Experimental Analysis of Distribution Center Vision-Based Truck Localization System

Chirascu, D.C.

Award date:
2020

Eindhoven University of Technology

Department of Mathematics and Computer Science

Model Driven Software Engineering

Department of Mechanical Engineering

Automotive Engineering Science (AES)

Experimental Analysis of Distribution Center Vision-Based Truck Localization System

Dan Cristian Chirascu

0923784

Supervisor:

Dr. Ir. Ion Barosan

Eindhoven, March 2020

Acknowledgements

I would first of all like to thank supervisor dr. ir. Ion Barosan for his support throughout the thesis. His continuous supportive supervision and positive attitude has been invaluable towards realising the current implementation and understanding the localization system.

I would like to thank dr. ir. Richard Verhoeven for his advice regarding the networking and computation aspects of the localization system. Furthermore, he enabled the implementation of the Raspberry-Pi localization system by providing the necessary equipment.

I would like to thank Dr. Ömür Arslan who provided advice concerning presentation and structuring of this work.

I would like to thank dr.ir. Igo Besselink for his supportive attitude and continuous effort for the realisation and improvement of the Trucklab as a whole.

I would like to thank my colleague Glenn Smeulders (MSc) for his support in understanding the marker pose estimation process.

Finally, I would like to thank my parents for their continuous support without which my masters degree would not have happened.

Abstract

Trucks are the predominantly used vehicles for land freight transportation in EU member countries (EU-27), UK and in the US. Companies involved in the production or use of trucks, as well as in the development of autonomous vehicle technologies, are looking towards automation of truck driving as one of the solutions for increased productivity and road safety. Part of this automation problem is the automated localization and maneuvering of trucks in the area of a distribution center loading dock. Based on experiments performed, this work describes the localization process of the Eindhoven University of Technology (TU/e) truck localization system and of vision-based localization systems in general. To characterize system performance, localization accuracy and execution time parameters are identified. Proposals are made for optimization of execution time for localization systems that employ distributed processing.

Acronyms

AGV Automated Guided Vehicle. 2

AV Audio Video. 42

DMA Direct Memory Access. 30, 32, 33, 39

gPTP generalized Precision Time Protocol. 40

HAN Hogeschool Arnhem en Nijmegen. 8, 26, 44

IoT Internet of Things. 10

OpenCV Open Source Computer Vision Library. 13–15, 18, 25, 27

SAE Society of Automotive Engineers. 5

TSN Time-Sensitive Networking. 40

TU/e Eindhoven University of Technology. 8, 9, 14, 38, 44

Table of Contents

Chapter 1

Introduction

1.1 Motivation

Logistics is generally defined as the detailed organization and implementation of complex operations [1]. In military science it is the process of planning and executing the movement and maintenance of military forces [2]. From a business perspective, it is the management of resource flow between source and origin in order to meet the requirements of customers or corporations [1]. The scope of this work is within the field of business logistics, more specifically the transportation of material resources on land via trucks. Freight can also be transported via rails, aircraft, naval transport - inland waterways or oversea. Regarding the use of trucks in the overall logistics landscape, US Bureau of Transportation Statistics [3] reports and projects that between 63% and 65.5% of freight by tonnage in the US (domestic, import and export) was and will be carried by trucks in the years from 2012 to 2045, as shown in figure 1.1. In the EU27 and UK, road transportation represents between 74.6% and 76.7% of inland freight transportation mode in tonne-kilometers [4], stats per year shown in figure 1.2. In 2017, 81.4% of EU27 and UK road freight transport was done by vehicles with a maximum permissible laden weight of over 30 tonnes [5]. These statistics indicate that trucks play an important role in material logistics on multiple continents.

Logistics automation is the application of computer software or automated machinery to improve the efficiency of logistics operations [6]. Automation can be implemented inside a warehouse or a distribution center, involving the use of static machinery such as cranes, conveyor belts, sorting machines or mobile robots such as an Automated Guided Vehicle (AGV). Different types of AGVs, each with a specific job to perform such as unloading trucks, co-packing, picking orders,

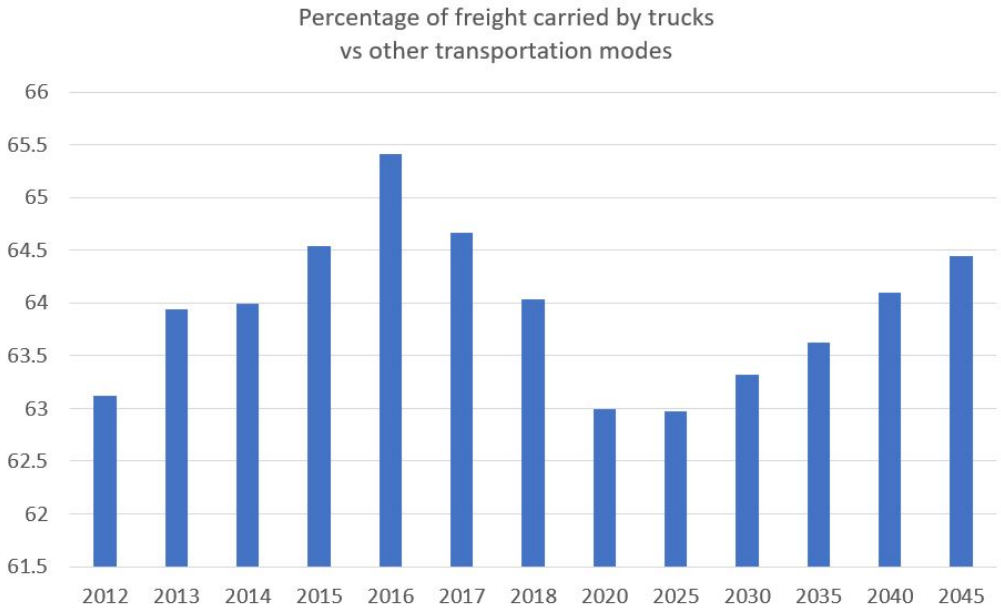


Figure 1.1: Percentage of freight by tonnage transported via trucks vs other transportation modes in US [3].

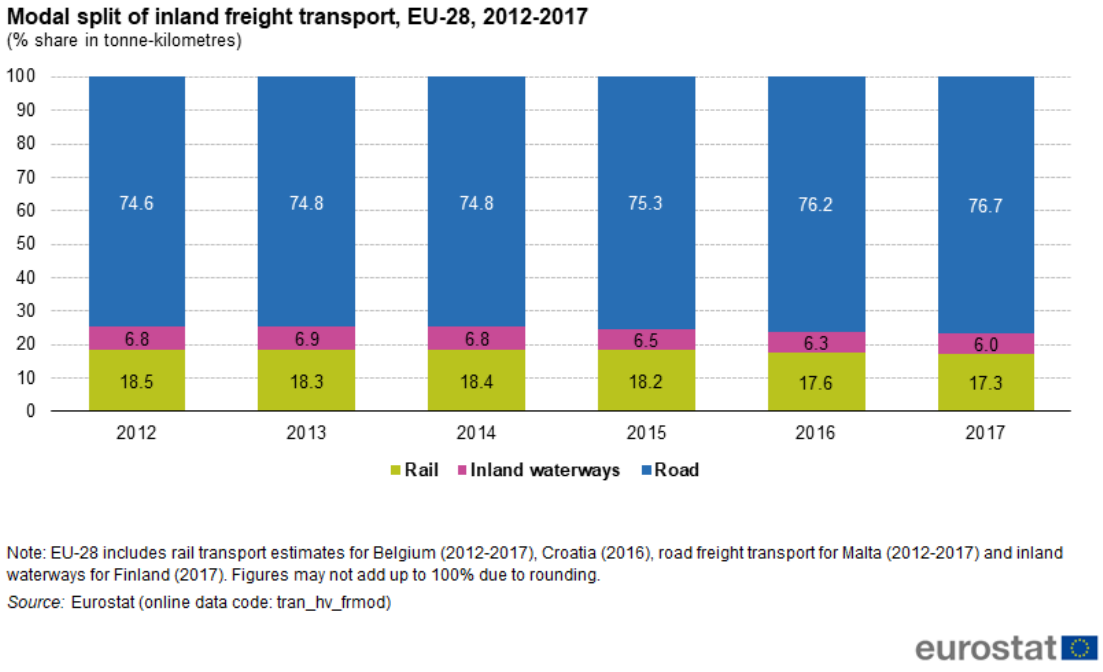


Figure 1.2: EU27 and UK freight - modal split [4]

checking inventory, or shipping goods. According to [7], in the future, most of these AGVs will be mobile and self-contained but they will be coordinated through distribution center management systems and equipped with planning software to track inventory movements and progress orders with a high degree of accuracy. A distribution center of the future is showcased in figure 1.3. Outside of storage facilities, automation can be implemented on delivery vehicles: drones, trucks, underground vacuum tube trains (SpaceX Hyperloop) [8].

Figure 1.3: Distribution center concept showcasing various automated vehicles [7]

Trucks play an important role in logistics and therefore truck automation is a crucial necessity for automation of logistics processes. Companies involved in production or use of trucks, or in the development of autonomous vehicle technologies are looking towards automation of trucks as one of the solutions to achieve:

- Increased efficiency - reduced fuel consumption and wear-and-tear.
- Productivity - round-the-clock operation, faster maneuvering and route planning.
- Safety - reduced blind-spot collisions, instability and loss-of-control accidents.

A 2019 Society of Automotive Engineers (SAE) lecture [9] enumerates truck automation projects in the US:

- US Department of Energy's ARPA-E (Advanced Research Projects Agency - Energy) funded projects to integrate data obtained through vehicle connectivity with external sources with the powertrain control system, to improve fuel economy.
- The University of Minnesota, Workhorse and UPS are looking to improve fuel economy of individual range extended hybrid delivery trucks by using cloud connectivity and data analytics.
- Pennsylvania State University, North Carolina State University, Clemson University and Volvo Group work on real-time, collaborative and predictive co-optimization of routing, speed and powertrain control in Class 8 in heavy 15 ton or more trucks. The project aims towards platooning, terrain predictive control and automation at road intersections.
- Purdue University, Cummins and Peloton aim to improve fuel consumption of class 8 trucks using connectivity and automation-enabled control systems.
- Daimler demonstrated two-truck platooning in June 2018, has brought SAE level 2 [10] (partially-automated) trucks into production and is aiming towards producing SAE level 4 automated trucks [11].

In Europe projects include:

- DAF, NXP Semiconductors, TNO (the Netherlands Organisation for Applied Scientific Research) and Ricardo are collaborating to achieve truck platooning [12].
- Multi-brand truck platooning by Ensemble [13].

These projects represent state-of-the-art development in truck automation, however, with reference to the SAE levels of vehicle automation, stay within Levels 1 and 2 where the driver is still required (see figure 1.4). In [9], the authors claim that full autonomy (SAE level 5) is long to come. Automated driving will take likely more than 10 years (as of 2019) to have a measurable market penetration, however first use cases are expected to hit the market within the next few years, platooning-enabled long-haul truck being the first to begin selling.

At the moment, truck automation is in its infancy [9] as only partial levels of automation have been achieved and only for linear motion on highways. Maneuvers within a distribution center loading dock are more complex, as they require the truck to maneuver in a more restricted, possibly

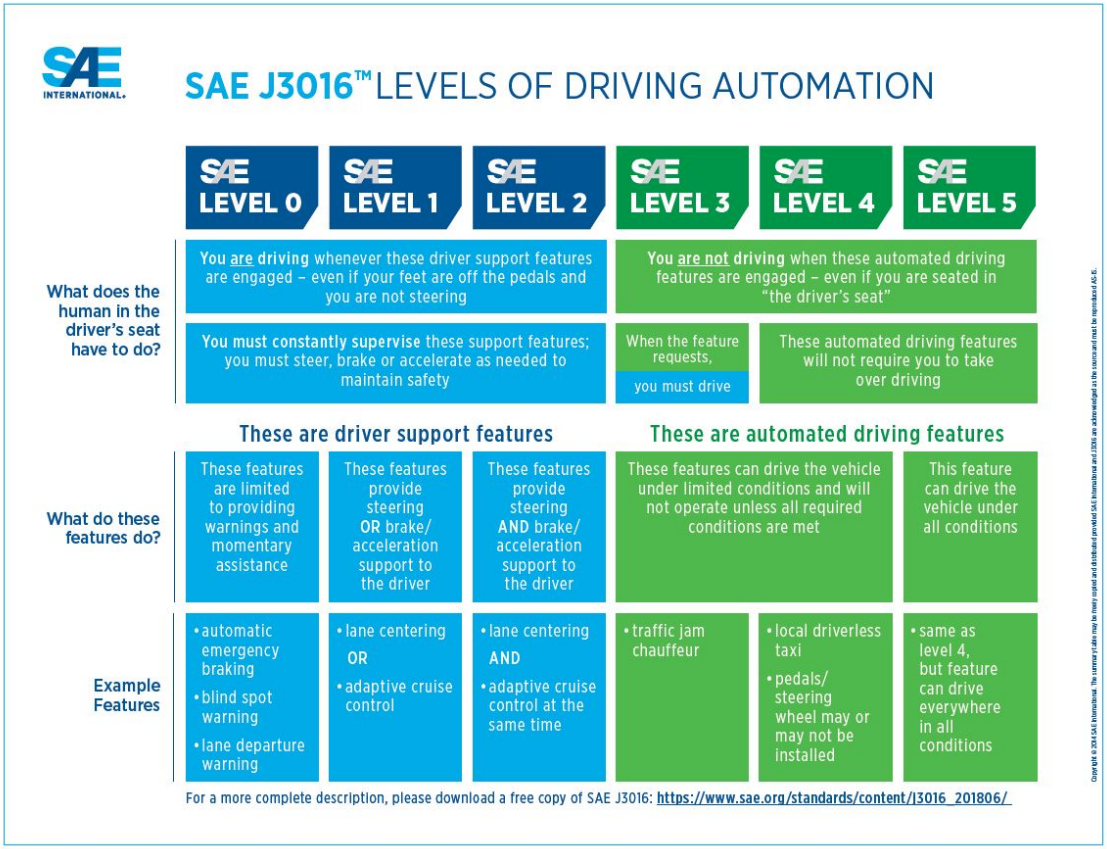


Figure 1.4: SAE Levels of Automation [14]

obstacle-prone area. Driver-assist technologies such as forward collision warning system (FCWS), blind spot detection system (BSDS), 360 LIDAR vision can assist drivers in their maneuvers. The focus of this work is however on autonomous trucks, which require accurate real-time localization to maneuver in a distribution center loading dock. Localization can be achieved in multiple ways:

- GPS - This method is already widely used to locate vehicles however it cannot achieve sub-meter accuracy unless some method of sensor fusion is employed. This is done in [15] where computer vision via on-board camera is combined with GPS. The method uses markers placed near the road and assumes these markers are always visible, which is not always the case in a distribution center loading dock.

- Odometry - These are minimalistic systems that measure rotation and velocity or wheel rotations to obtain distance and location. They are however error-prone due to small errors building up over time [16]. [17] use error-prone odometers and compensate for errors with vision-sensors.

- Radio frequency - In [18], an RSSI (Received Signal Strength Indicator) approach is taken

using Wi-Fi access points. RSSI values are however heavily affected by Wi-Fi access point location with respect to the detected object. [19] uses RSSI based trilateration, requiring further processing done by ANNs (artificial neural networks). [20], [21] and [22] use a 802.15.4 network to localize AGVs. [23] describes how RFID (Radio Frequency Identification) can be achieved by using a RFID transponder embedded floor.

- Laser - [20] employs laser range finders to compensate for the insufficient accuracy of WSN localization during docking maneuvers.
- Vision - The current work along with [24] describe how vision-based localization can be achieved using artificial markers. Localization without using artificial markers has been achieved in [25], where images taken by a robot are used to determine the robot location based on a global map of images.
- Fusion - Multiple category sensor fusion can be seen in [26], which achieves localization based on input from camera as well as Wi-Fi and laser sensors. [27] uses both error-prone inertial sensor as well as visual input to correct the former.

In an environment where multiple vehicles and various obstacles can be present, sensor signal interference, out-of-range and occlusion situations can occur. Furthermore, even though such technologies can be, and some are, used for obstacle detection and localization of smaller AGVs, trucks are harder to be detected in their entirety because of their size, and of being composed of a tractor and at least one trailer, both for which the pose must be determined. Ideally, the localization system will be able to estimate the pose of all trucks in the loading dock in real-time and also detect other moving/stationed entities for hazard mitigation. For this, a sensing system that does not suffer from interference or occlusion situations and can capture an entire truck within sensor range is needed. A system using vision (with or without other sensors) is more likely to fulfill such requirements, as cameras can be placed in the area above the truck maneuvering space so that the required area can be surveyed. Tractor and trailer pose can be estimated by processing the visual data, either with or without artificial markers, an example of the latter being detection of a truck's shape.

In spite of these advantages, vision-based localization systems can still suffer from inaccuracy. Furthermore the system may be required to localize under certain real-time requirements to avoid maneuvering trucks incorrectly. Given these issues, this work aims towards:

- Identifying the system parameters that influence accuracy and execution time.
- Exploring how a localization system can be designed and how architecture design choices can influence system parameters.

- Describing the localization process of the system implemented within TU/e and of any vision-based system in general.

Before the problem is stated, the prior work found will be be described.

1.2 Prior Work

No prior work has been found with regard to localization of trucks within a distribution center loading dock, with the exception of the work done within Hogeschool Arnhem en Nijmegen (HAN) [24], where the localization method proposed is also vision-based and uses ArUco markers. The pose estimation methods used within HAN [24] will be taken in consideration when discussing the localization process and accuracy related parameters (Ch. 3).

The work done within TU/e in [28], [29], [30], [31] and [32] involve truck kinematic and path planning models, truck maneuver virtualization and truck remote control. Little to no attention is given to the localization system. The pose estimation process is yet to be understood and the execution time of the truck pose information is yet to be measured. The parameters affecting pose estimation accuracy and execution time are also not discussed.

Oma et. al [33] [34] describe an abstract fog computing model that quantifies the size of the data sent by sensors, its reduction as a result of the processing done by fog nodes, as well as the energy and power consumption required for data storage and processing. Oma et. al focus on analysing the impact of data reduction on overall energy consumption. The analysis in this work will instead focus on the impact of data reduction on overall execution time, as energy consumption of the system is not in the scope of the current work.

Given this current state of research into the problem of truck localization the problem is stated.

1.3 Problem Statement

Truck maneuver automation is required to help mitigate hazards and improve the efficiency and productivity of logistics operations. This includes the automation of truck maneuvers in the area of a distribution center, which requires a localization system that detects the truck pose while satisfying real-time and accuracy requirements. To model, validate and build such a system, understanding of the pose estimation process, critical parameters and architectural choices is required.

1.3.1 Research Questions

1. How is the truck pose estimated by the TU/e Trucklab localization system and in general?
2. What are the system parameters that influence accuracy?
3. What are the system parameters that influence execution time?
4. How will a decentralized system perform in comparison to the current centralized system?

1.4 Organization of the Report

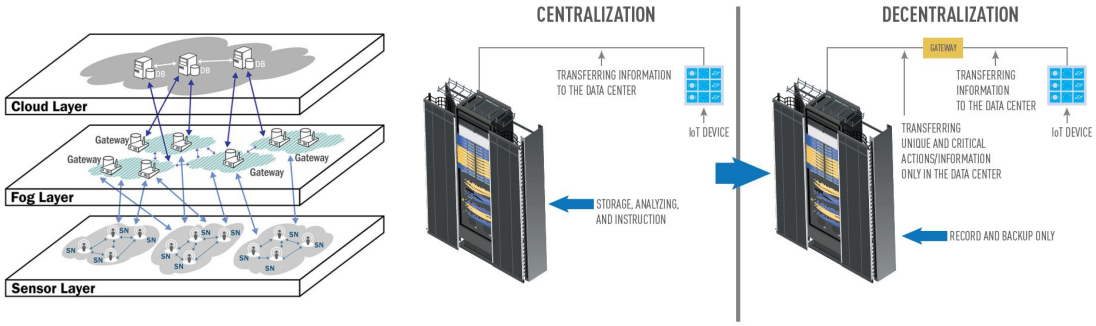
In chapter 2 the architectural choices and network protocols are discussed and the implemented localization systems are described. Chapter 3 describes the experimental analysis of the system in terms of accuracy and execution time. Finally, the research questions are answered in chapter 4 and proposals for improvement are given.

Chapter 2

Localization Approach

When designing a localization system, one must take into consideration both the localization method and the possible ways to implement it. This chapter focuses on describing what choices can be made regarding architecture and network protocols, the method and tools used as well as the current system implementation.

2.1 Architectural Choices



(a) Fog layer

(b) Centralized vs decentralized processing

Figure 2.1: Fog layer and decentralized processing [35] [36]

A localization system is comprised of a network of sensing and processing elements, that may in turn be part of a larger network. The Internet of Things (IoT) paradigm can provide insight as to what kind of network a localization system could be based on or part of. IoT is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction [37]. It involves a high degree of

decentralization of processing and data storage as opposed to the centralized (cloud) architectures.

Machine-to-machine communication for example enables companies to facilitate work-flows in various departments: manufacturing for controlling factory floor activity; logistics, for tracking warehouse inventory; supply chain for linking with suppliers to track raw materials and finished goods [38]. Recall figure 1.3 illustrating the distribution center of the future, where AGVs are operating in an interconnected way in order to perform distribution center operations. The localization system may be implemented as part of a network consisting of many interconnected things.

The introduction of a large number of connected devices requires a scalable architecture to accommodate them without any degradation of the quality of service demanded by applications [35]. This can be achieved by extending the functions of centralized (cloud) computing closer to the end devices via an intermediate layer known as the "fog" (see figure 2.1a). In the fog approach, computing is done by multiple devices and possibly also network gateways. A distributed structure allows for more options with regards to the system functional configuration in terms of where data is processed, communicated, stored or discarded. A possible approach is sending only unique and critical information to the central computer, which may allow for better real-time performance, as shown in figure 2.1b. An example functional view of a decentralized system data pipeline can be seen in figure 2.2. When designing a localization system, there are multiple choices that can be made regarding its architecture:

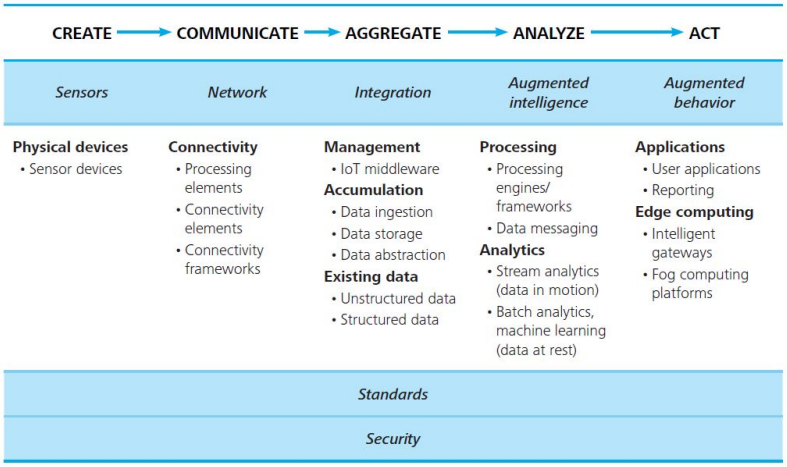


Figure 2.2: Example IoT Network Data Pipeline [39]

- Structural choices - number and location of sensors; processing and storage nodes, how these are linked.

- Functional choices - which data is processed or stored, where is data processed or stored, communication protocols.

2.1.1 Network Protocols

Naturally, for data to be communicated throughout the localization system network, a communication protocol is required. In the scope of this work the protocols FireWire , Ethernet and Wi-Fi will be described and compared. FireWire and Wi-Fi are protocols used in implementations as part of this work. Ethernet is also described since it is the most widely used protocol, along with Wi-Fi, for Local Area Networks (LANs) and one of the most used protocols in industrial [40] and IoT applications [38], [39], [36], [41]. A summarized comparison of the three protocols can be seen in 2.1.

Metric/Protocol	FireWire	Ethernet	Wi-Fi
Max bandwidth	800 Mbps (most common) 3.2 Gbps (specified)	10 Gbps (twisted-pair) 100 Gbps (fiber-optic/coaxial)	72–600 Mbps (Wi-Fi 4) 433-6933 Mbps (wi-Fi 5) 600–9608 Mbps (Wi-Fi 6)
Max range (1 hop)	8m	100m (twisted-pair)	46m Wi-Fi 5 77m Wi-Fi 6
Topology	Bus	Tree	Tree
Scalability	Max 64 devices	Virtually limitless	Virtually limitless (with no channel collisions)
Transmission	Broadcast only (bus)	Point-to-point full-duplex	Broadcast half-duplex

Table 2.1: Data transfer protocol comparison

IEEE 1394 - FireWire: The first FireWire standard [42] introduced speeds up to 400 Mbps. In 2002 [43] the technology allowed for 800 Mbps and, further on, in 2008, specifications were made for 1600 and 3200 Mbps. In the same year however, the protocol was no longer used by Apple, the company who commercially trademarked it as "FireWire ". As of today, few products still use FireWire as Ethernet is becoming a universal protocol for networks in general [44].

IEEE 802.3 - Ethernet: Many application-specific networks use older, serial, synchronous networks. The trend in device industry however is towards standardization on wireless or cable Ethernet based industrial protocols [38]. The twisted-pair 10 Gbps (10GBASE-T) Ethernet standard was developed and published in 2006 [45]. Even higher speed technologies exist: standardized in 2010 [46], 40/100 Gbps Ethernet is available for fiber optic cables and short-range copper coaxial cables [47]. Due to being a tree topology, packet-switched network, it can be increased virtually indefinitely. Twisted-pair cable Ethernet can reach up to 100m without the need for a hub [48], which is far better than any other major bus used in the machine vision industry. It furthermore allows for easy interoperability with other networks, local or from across the internet.

IEEE 802.11 - Wi-Fi: Designed to seamlessly inter-operate with its wired sibling Ethernet, the first version of the 802.11 protocol was released in 1997, and provided up to 2 Mbps link speeds. This was updated in 1999 with 802.11b to permit 11 Mbps link speeds, and this proved popular.

In 1999, the Wi-Fi Alliance formed as a trade association to hold the Wi-Fi trademark under which most products are sold. The Wi-Fi alliance standardized generational numbering so that equipment can indicate that it supports Wi-Fi 4 (if the equipment supports 802.11n), Wi-Fi 5 (802.11ac) and Wi-Fi 6 (802.11ax). It can operate within 2.4Ghz spectrum, 5Ghz or, in case of Wi-Fi 6 between 1 and 6 Ghz. Channels can be shared between networks but only one transmitter can locally transmit on a channel at any moment in time (half-duplex transmission) [49]. Max range for Wi-Fi 5 is 46 m. Wi-Fi 6 can reach 77m in indoor occlusion [41].

With regards to scalability, the most scalable of the three is Ethernet , as FireWire operates on a single bus and Wi-Fi only provides half-duplex transmission on a shared channel. FireWire however is no longer under development and Ethernet and Wi-Fi standards and technologies further seek to improve data rate, predictability and thus overall scalability of the protocol. Scalability can be achieved by spatially separating same frequency nodes or by Carrier-sense multiple access with collision avoidance (CSMA/CA). Wi-Fi 6 however has brought significant contributions to scalability as explained below. In the particular case of Wi-Fi, the 6th generation Wi-Fi standard, IEEE 802.11ax, allows for better scalability, data rate, real-time performance via the following new technologies:

- Higher data rate is achieved by increasing the Quadrature Amplitude Modulation (QAM) from 256 (used in Wi-Fi 5) to 1024. This adds two extra bits per symbol thus increasing throughput by 25% (for more on QAM see [50]). Higher wireless throughput facilitated by 1024-QAM is critical to ensuring Quality-of-Service (QoS) in high-density locations such as stadiums, convention centers, transportation hubs, and auditoriums [50]. Within our scope it worth mentioning that, through increase in transceiver bandwidth and QAM, the standard aims for Wi-Fi to support of 4K/8K video streaming, which is becoming the video quality norm [50], [41].
- Higher predictability is achieved by Wi-Fi 6's OFDMA channel access mechanism which allows for bandwidth to be allocated according to a client's needs, as opposed to Wi-Fi 5's fixed frequency OFDM (for more see [51], [41]).
- Flexible low-power device scheduling for better power efficiency is achieved via OFDMA with Target-Wakeup Time (TWT) (see [41]).

2.2 Localization Tools

Implementation wise, truck localization is done using fiducial markers - more specifically ArUco markers [52] - and the Open Source Computer Vision Library (OpenCV) library. This choice

has been made because this work makes use of an already existing implementation in the TU/e Trucklab based on pose estimation of such markers with OpenCV ArUco library. This section gives an introduction to fiducial markers and the OpenCV library.

2.2.1 Fiducial Markers

Fiducial markers are artificial landmarks that facilitate correspondences between image points and reference real-world coordinates. Fiducial marker systems consist of some unique patterns along with the algorithms necessary to locate their projection in camera images. Such a system is designed to provide a list of markers found in the image, given an input image, either static or a frame from a video stream. The extracted information can be used in different ways, such as triggering a certain behaviour upon marker detection or estimating the marker pose.

Before discussing fiducial marker systems, it is useful to mention that bar codes are not part of this category. They are intended to carry information, not to localize, as is needed as a fiducial for pose estimation applications. Other bar codes are DataMatrix, Maxicode and QR. They are not as useful for localization as fiducial marker systems, as they do not provide enough image points for 3D pose calculation and typically require a large area in the image, limiting the range at which they can be used.

Simple fiducial markers, that carry less information content than 2D bar codes, are simply dots and discs, for which the fiducial marker system finds the center. Related ideas are systems that use flat circular dot or checker-board patterns. Dots can be expanded to have multiple IDs by enclosing the in rings. There are more ways to enclose data can be enclosed in circular boundaries, however these can only provide a high degree of accuracy for a single point, while such systems provide low accuracy in determining the pose from

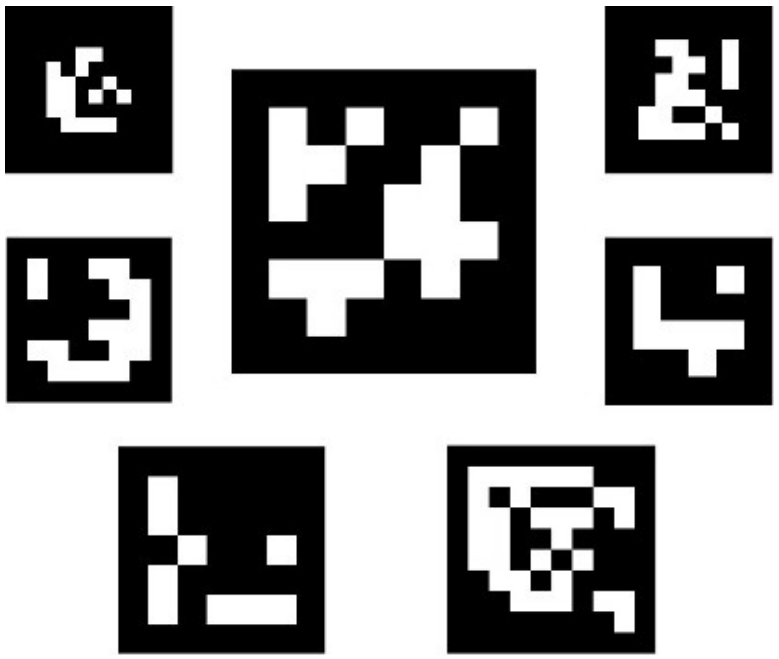


Figure 2.3: Sample ArUco markers [53]

a single marker. Square shapes however provide four salient points, which is useful both for computing pose from a single marker and for decoding the markers [54].

The main advantage of square shaped markers is that the presence of four prominent points can be employed to obtain the pose, while the inner region is used for identification [52]. Examples are discussed in [55]. The author compares fiducial marker systems ARTag, AprilTag and CALTag, with respect to resistance in case of occlusions, the latter being most resistant. In [52] the ArUco marker system is presented, which is similar to ARTag, only has better detection in case of occlusion. Figure 2.3 shows sample ArUco markers.

2.2.2 OpenCV

OpenCV is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products [56]. The system under study uses the OpenCV ArUco library to detect and and estimate the pose of ArUco markers.

2.3 Localization Method

The localization system uses ArUco markers in order to estimate the pose of the truck tractor and trailer. The input of the system is the pixel data from multiple cameras covering the distribution center loading dock. The output is the estimated pose of the truck tractor and trailer with respect to the global reference point (point $(0, 0)$ in figure C.1). There are two types of ArUco markers used:

- Moving markers: these markers are placed on the truck. More exactly, one above the front axle of the tractor and one above the rear axle of the trailer. Two markers are used so that the coordinates of both the tractor and trailer with respect to the origin can be estimated, as well as the angle between the two. The moving markers have a side length of 13 centimeters.
- Fixed markers: these markers are used as intermediate reference points in order to determine the global pose. They are fixed to the floor at known coordinates with respect to the global reference. Without them, only the pose in the local camera reference can be estimated. The pose estimation process is explained in detail in chapter 3. The fixed markers have a side length of 20.3 centimeters.

For localization to be possible, the camera frame used must contain at least one moving and one fixed marker. Figure 2.4 shows a simplified representation of fixed markers placed on the floor

and moving markers on the truck. Each camera is positioned above ground, facing down, so that there is at least one fixed marker within the camera range. There are two camera networks used: a network of Raspberry-Pi cameras and one of FireWire cameras. They are both used independently and no sensor fusion between a Raspberry-Pi and a FireWire camera is done in the current implementation.

If only one camera was to be used, covering the entirety of the distribution center area, fixed markers (used as intermediate references) would not have been necessary. For an implementation where only one camera that covers the entire loading dock area is used see [57], where there are no fixed markers used as intermediaries, but only as destination points for the trucks. In our case however, one FireWire or Raspberry-Pi camera alone does not cover the entire distribution center area and as such the global position of the moving markers is determined using the fixed marker as intermediary.

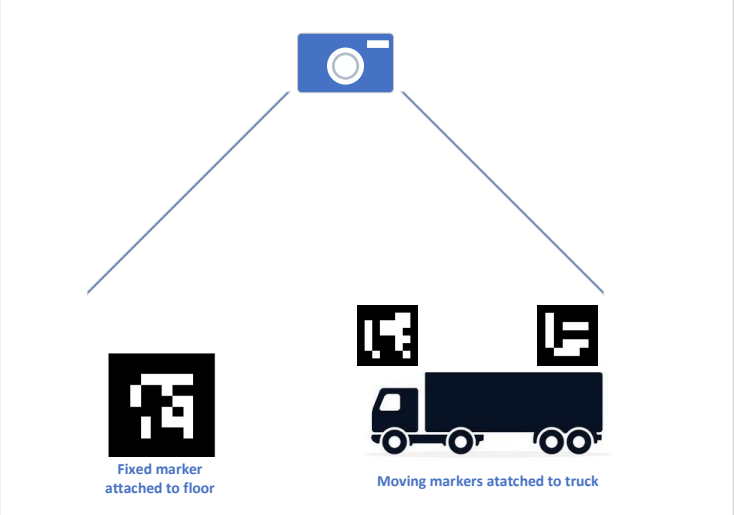


Figure 2.4: Moving markers are localized in a frame with respect to the location of that frame's fixed marker

2.4 System Implementation

Given the above descriptions of the architectural choices and localization method, the implemented system is now described in terms of component structure and component functions.

2.4.1 Physical Components

The localization system is implemented within the Automotive Engineering Systems (AES) Truck Lab. This is an environment where students and engineers can develop automated tractor semi-trailer driving functionalities for use in the area of a distribution center. The localization system is formed of the following components:

- A scaled-down autonomous truck. The automation of the truck is realized using the TurtleBot3 Waffle Pi platform. The Turtlebot specifications can be seen in [58]. The truck has

two ArUco markers fixed on top: one above the front axle of the tractor and one above the rear axle of the trailer. These markers will be referred to as moving markers.

- A scaled-down model of a distribution center docking area, which is a replica of the Jumbo distribution center in Veghel, Netherlands. Distribution center docking area dimensions can be seen in appendix C.
- A set of 8 machine-vision cameras communicating to the central computer via FireWire . Their specifications can be seen in appendix A.
- 8 markers fixed on the floor in the field of view of the 8 cameras. These markers will be referred to as fixed markers. Each FireWire camera can detect only one fixed marker.
- A set of 6 Raspberry-Pi 's each equipped with Pi-Camera v2.1. This work uses the Raspberry Pi 3 Model B. The device specifications [59] can be found in appendix B. Specifications for camera can be found in [60].
- A central computer running Ubuntu Linux with:
  - Intel® Xeon(R) CPU E5-2620 v4 @ 2.10GHz processor and
  - LSI Corporation FW643 [TrueFire] PCIe 1394b Controller (rev 08).

2.4.2 System Layout

The FireWire cameras and ArUco markers are placed as shown in figure 2.5, with each camera being able to detect one fixed marker. The cameras together cover the distribution center loading dock. Notice that there is some overlap between a camera and its neighbors, as shown by the duplicated yellow rectangles in the middle. The Raspberry-Pi cameras are placed in a similar fashion, with the exception that there are only 3 cameras instead of 4 per row. This is due to Pi cameras being able to cover more area than the FireWire cameras. Some of the Pi cameras can spot two fixed markers at once. In such situations, one of the markers was covered.

2.4.3 Network Architectures

Currently, there are two network architectures implemented:

- The centralized system uses 8 FireWire cameras to cover the distribution center area. The cameras are connected to a central computer via a common FireWire bus. The cameras send the frame data to the main computer which processes it, after which the estimated pose information for each moving marker is obtained. Figure 2.6a shows a diagram of the centralized network.

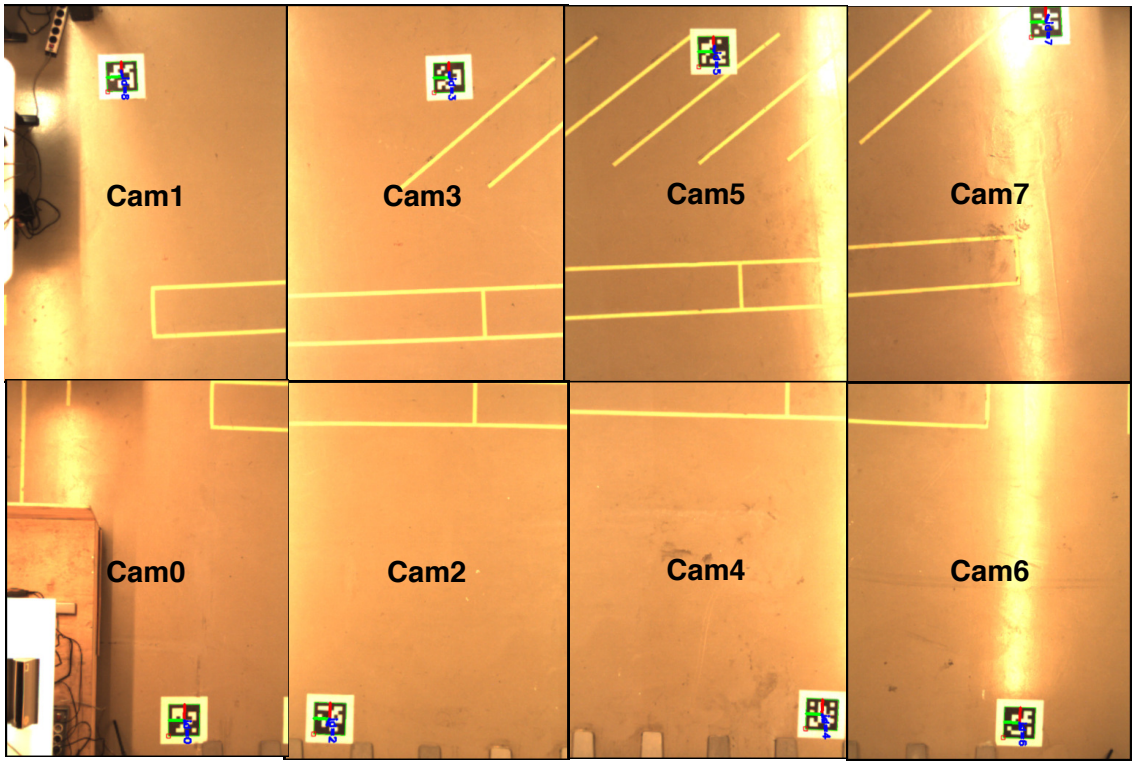


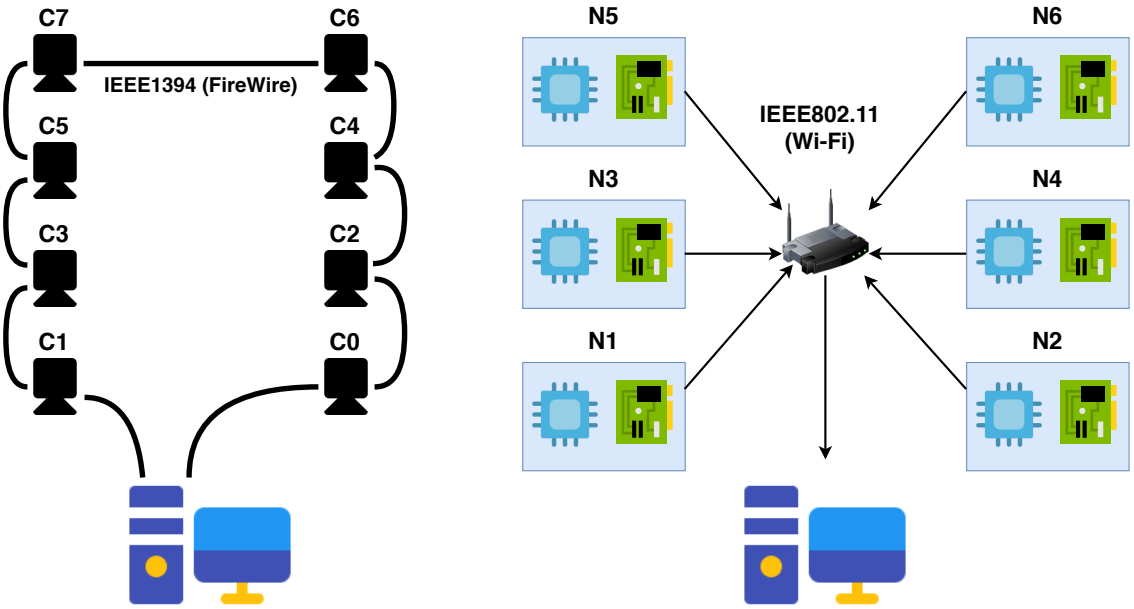
Figure 2.5: FireWire camera and ArUco markers layout

- The distributed system uses 6 Raspberry-Pi and Pi camera pairs to cover the distribution center area. The Pi cameras are connected via a Camera Serial Interface (CSI). Each Raspberry-Pi receives the frame data from its associated Pi camera and processes it. The obtained pose estimation values are sent towards the main computer to be aggregated and fused into a single pose value per moving marker. Figure 2.6b shows a diagram of the decentralized network.

2.4.4 Software Implementations

The software implementation for both FireWire and Raspberry-Pi camera networks is done in the `C/C++` programming language and uses the OpenCV ArUco library to estimate the pose of ArUco markers. There are multiple code implementations written. As part of this thesis, two implementations have been used for the centralized system and one for the decentralized system. They are described below:

- Implementations for the centralized system:
  - "Centralized-Sequential": the central computer receives all frames from the FireWire cameras and sequentially identifies and estimates the pose of moving markers present in the



(a) Centralized network **(b)** Decentralized network

Figure 2.6: Implemented network architectures

frames. For every frame, if there is a moving marker for which the pose has been estimated in a previous frame, the pose information for that frame is skipped. This is done in order to avoid sending pose estimation information for one marker from separate frames, which can result in inconsistent pose information.

  - "Centralized-Threaded": the central computer assigns the receiving and processing of the FireWire cameras to threads, with one thread for each camera. The estimated poses are fused using averaging. Thus if the same moving marker is present in multiple frames, its estimated pose values are averaged and the result is used.

- Implementation for the decentralized system:

  - "Distributed-Processing": The Raspberry Pi's receive frames from Pi cameras attached to them and extract the marker pose from the frame data. The pose information is then sent to the central computer which performs sensor fusion by averaging.

In the following chapters, a more in-depth analysis of the localization process and system behaviour is done so that the relevant parameters influencing accuracy and execution time can be identified. Furthermore, this will provide understanding of OpenCV-based as well as general vision-based localization processes. Finally, the architectural choice influence on execution time is also analyzed.

Chapter 3

Experimental Analysis

3.1 Parameters Overview

A DC truck localization system has one fundamental purpose, namely to answer the question "where is the truck now?" and to do so in an acceptable amount of time. The analyzed system performance is thus characterized by the accuracy of the pose information reported and the time in which it is reported. The accuracy and execution time parameters are further split as shown in figure 3.1.

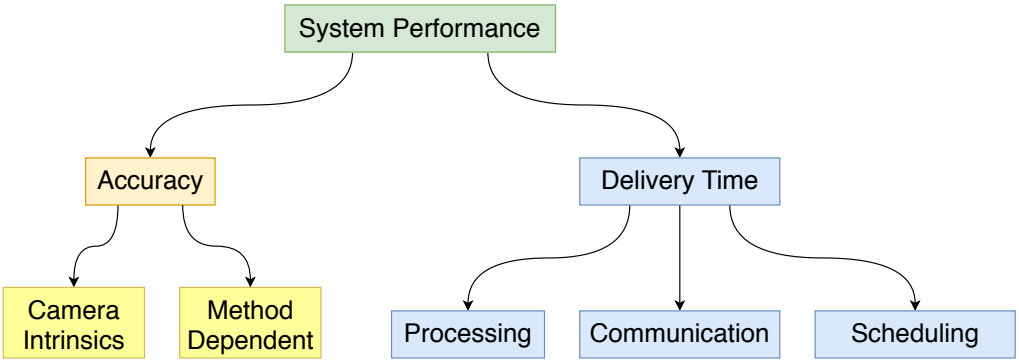


Figure 3.1: System parameters pertaining to accuracy and execution time

3.2 Localization Process

In any vision based localization system there are cameras which capture real world images and transform them into pixel data. This pixel data is then acquired by the localization system which turns it into information necessary to localize the trucks relative to the reference system used. The first factor in the data pipeline that influences accuracy are the camera intrinsic parameters.

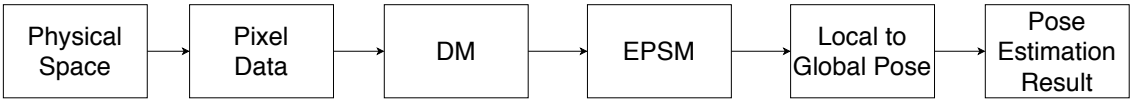


Figure 3.2: OpenCV pose estimation process - full dataflow

These influence the pixel to physical-unit conversion of the pixel points of interest. For this thesis, processing using OpenCV functions is described, however, regardless of processing, if there are inaccuracies caused by lens distortion, then those inaccuracies will be propagated throughout the data pipeline. As such, camera intrinsics are a general parameter for any vision-based localization system. In order to determine what parameters form the camera intrinsics, as well as what other implementation-specific parameters influence accuracy, the OpenCV ArUco marker pose estimation process will be explained. For more information on the functions described and the processing done by OpenCV aruco module in general see [61], [62], [53], [63], [64].

figure 3.2 shows the pose estimation process done in the current implementation. Once the frame (pixel data) is acquired, the first processing step is the `detectMarkers` function, abstractly represented as `DM` in figs. 3.2 and 3.3. The purpose of this function is to detect markers in the provided frame and output the marker IDs (from the dictionary) and the coordinates of each marker's 4 corners.

The `DM` function takes as input the following parameters:

- frame data (`frame`): frame pixel data on which marker detection is performed;
- marker dictionary (`dict`): the dictionary that maps a marker pattern to an ID;
- detection parameters (`dP`): various parameters that can be used to customize the detection process.

The following parameters are returned as output:

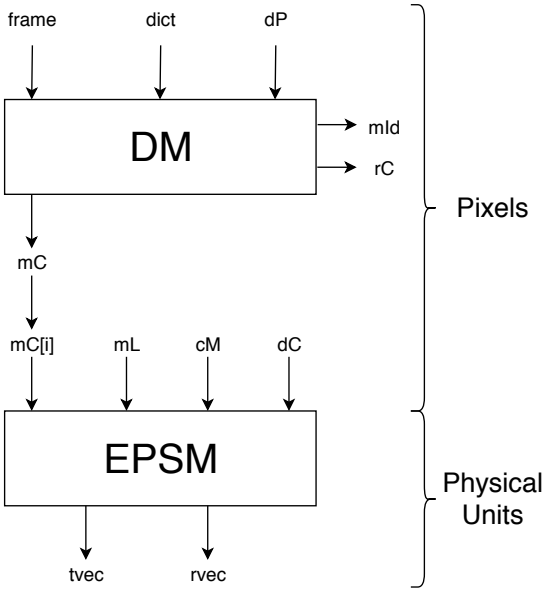


Figure 3.3: OpenCV pose estimation process - DM and EPSM

- detected marker IDs (`mID`);
- marker corners (`mC`): marker corner points in (x,y) float-value pixel coordinates. This parameter is an array where each element is the 4-corner set of each detected marker. The order of the corners is clockwise;

- rejected marker candidates (`rC`): shapes that were found and considered but did not contain a valid marker.

In order for the marker pose to be estimated, the marker corners must be forwarded to the `estimatePoseSingleMarkers` function, represented abstractly as `EPSM` in figs. 3.2 and 3.3. This function uses the marker corners, the marker length and camera intrinsic parameters in order to output the rotation and translation vectors (which are explained by ArUco paper author in [64]) of the marker relative to the camera. The right side of figure 3.3 indicates when that the marker data is initially in pixels (marker corners). It is `EPSM` that outputs rotation and translation vectors using the unit that the marker lengths are provided in, which can be any unit the user chooses.

The `EPSM` function takes as input the following parameters:

- a single set of 4 marker corners, shown as `mC[i]`, since it estimates the pose of a single marker;
- marker length (`mL`): the length of the side of the marker;
- camera intrinsic parameters: camera matrix (`cM`) and distortion coefficients (`dC`).

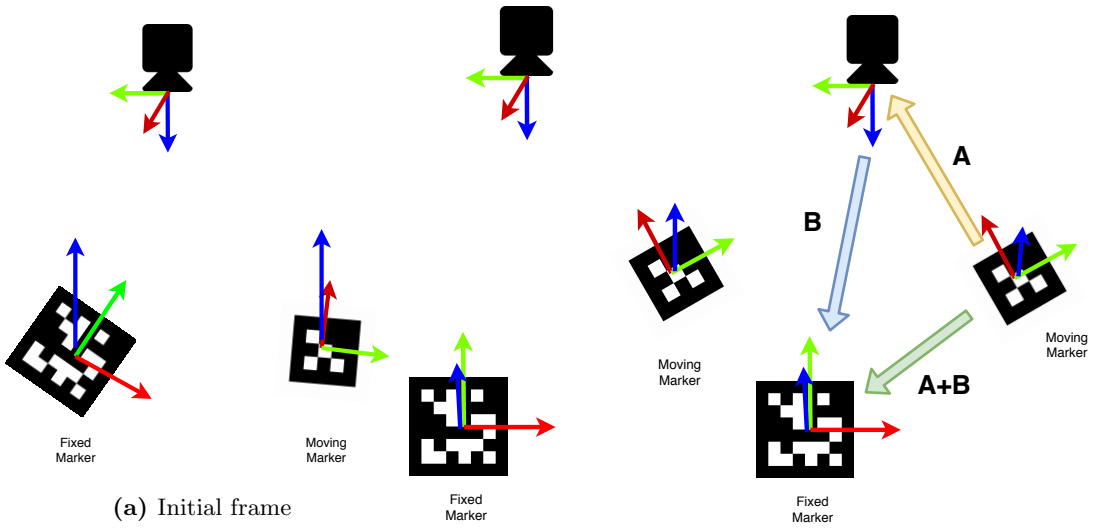
The following parameters are returned as output:

- translation vector (`tvec`);
- rotation vector (`rvec`) in Rodrigues format.

The translation and rotation vectors are used to transfer the coordinates of a point from the marker to the camera's reference. These are the marker pose parameters that form the marker pose information with respect to the camera reference system. The `tvec` and `rvec` of the moving and fixed markers in a frame are then used to calculate the global pose of the moving markers.

In order to determine the global pose of a moving marker, the system first determines its pose relative to a fixed marker, whose pose is known relative to the global reference. For this purpose, the rotation and translation vectors obtained by `EPSM` are useful because they allow the system to determine the pose of a marker relative to another marker. In the implementation used, the following calculation is used to determine the pose of the moving marker with respect to the fixed marker:

$$\texttt{tvec}_{m2f} = \mathcal{R}_f^t \cdot (\texttt{tvec}_m - \texttt{tvec}_f) \tag{3.1}$$



(a) Initial frame

(b) After rotation (multiplying with $\mathcal{R}_f^t$)

(c) Adding translation vectors

Figure 3.4: Obtaining pose of moving marker with respect to fixed marker. X is red, Y is green and Z is blue.

The rotation matrix $\mathcal{R}_f^t$ is obtained by applying the OpenCV `Rodrigues` function (which transforms a rotation vector to a rotation matrix or vice-versa) to $\texttt{rvec}_f$ (which is the rotation vector of the fixed marker with respect to the camera) obtaining the rotation matrix of the marker with respect to the camera $\mathcal{R}_f$. This matrix is then transposed, obtaining $\mathcal{R}_f^t$. The transposed matrix is necessary to rotate the camera reference so that it is aligned with the fixed marker reference system (Y pointing up and X to the right, as shown in figure 3.4b). A `tvec` is a vector that translates a point from marker to camera reference and so the negative of that `tvec` will do the opposite. To obtain the vector from moving to fixed marker reference ($\texttt{tvec}_{m2f}$ or A+B in figure 3.4c) we must add the `tvec` of the moving marker (A) with the negative of `tvec` of the fixed marker (B). We thus add a vector translation from moving marker to camera reference and from camera to fixed marker reference, therefore from moving marker to fixed marker reference.

In order to determine the coordinates of the moving marker with respect to the origin point `E0`, the already known translation and rotation between fixed marker and `E0` is used. Equation 3.1 is expanded to:

$$\texttt{tvec}_{m2\texttt{E0}} = \texttt{tvec}_{\texttt{E0}} + \mathcal{R}_{\texttt{E0}} \cdot \mathcal{R}_f^t \cdot (\texttt{tvec}_m - \texttt{tvec}_f) \tag{3.2}$$

Similarly as before, we rotate and add another vector. The rotation is necessary in order to align the fixed marker reference with that of the global reference of the DC loading dock. The rotation

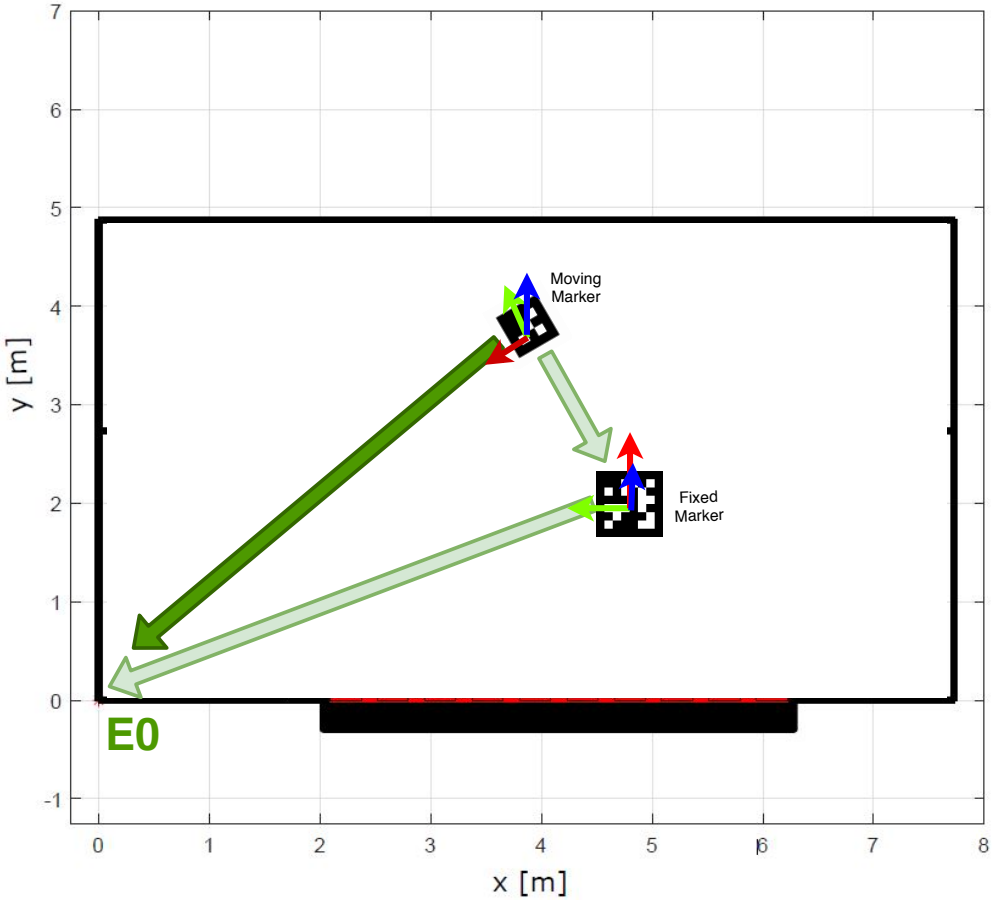


Figure 3.5: Obtaining pose of moving marker with respect to `E0`

matrix $\mathcal{R}_{\texttt{E0}}$ ensures that the X-axis of the fixed marker reference will point up, as that is the way the fixed markers are placed in physical space with respect to the loading dock reference (the X-axis of the fixed markers is in same direction as the Y-axis of the loading dock). The translation vector $\texttt{tvec}_{\texttt{E0}}$ is a vector based on physical measurements of the fixed marker locations with respect to `E0`. After rotating, $\texttt{tvec}_{\texttt{E0}}$ and $\texttt{tvec}_{m2f}$ are added (shown in figure 3.5 as the two light green vectors) in order to obtain the final vector (dark green) expressing the pose of the moving marker with respect to the global reference `E0`, which is the final translation vector sent towards the trucks.

3.3 Accuracy Parameters

It is necessary to go more in-depth regarding the `EPSM` function to find out what parameters influence accuracy in the current implementation. In figure 3.6 shows the `EPSM` function with the same inputs and outputs as in figure 3.3, only now two sub-functions are introduced and the transition from pixel data to real-world units is to some extent shown. First, there is the `_getSingleMarkerObjectPoints` (`GSMOP`) function, which takes the marker length parameter as input and outputs the coordinates of the 4 marker corners, in real-world units, with respect to the marker reference system. These are calculated as: `(-markerLength/2, markerLength/2, 0)`, `(markerLength/2, markerLength/2, 0)`, `(markerLength/2, -markerLength/2, 0)`, `(-markerLength/2, -markerLength/2, 0)` [61]. The first corner is the top left corner, followed by the top right, bottom right and bottom left. Finally there is the `SolvPnP` (`SPnP`) function which returns the transformation vectors, which are used as explained in section 3.2.

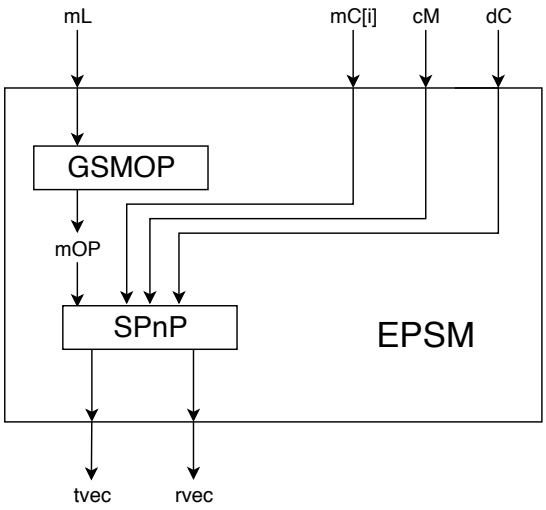


Figure 3.6: OpenCV marker pose estimation function

It is within the `SPnP` function that the camera matrix and distortion coefficients are used in equations that to obtain the estimated `tvec` and `rvec`. The equations are shown in [62] and they will not be further described as part of this work, as optical camera intrinsics are out of scope. Given the study of pose estimation done by the OpenCV ArUco library, The parameters that affect the accuracy of the current implementation are enumerated:

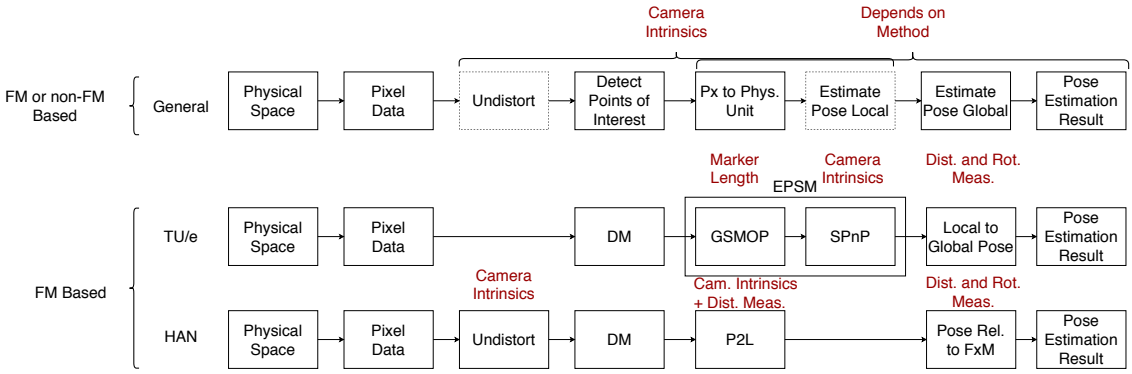


Figure 3.7: Generic vision-based localization model

- Camera intrinsic parameters:
  - camera matrix (`cM` - described in [62]);
  - camera distortion coefficients (`dC` - described in [62]).
- Physical space parameters, which are influenced by accurate measurement and placing of fixed markers and global origin point in physical space:
  - marker lengths (`mL`);
  - distance of marker center to global reference origin ($\texttt{tvec}_{\texttt{E0}}$);
  - rotation of marker reference from global reference ($\mathcal{R}_{\texttt{E0}}$).

Even if a different localization approach was used, the camera intrinsics remain parameters that influence accuracy. This is because, of course, any vision-based localization system requires cameras. After the pixel-data is obtained, this may or may not be undistorted and, furthermore, the operations performed to localize points in physical world differ according to implementation, each introducing its own accuracy-related parameters. This is shown in figure 3.7, where the implementation used by HAN [24] is also given as an example. In their case, undistortion is done before marker detection and the moving to fixed marker estimation is done using pixel to length conversion (`P2L`) with spacial calculations to determine moving marker pose relative to fixed marker (`FxM`). The `P2L` function varies for every camera used, as it depends on the camera intrinsics. To generate the function however, both pixel coordinates and physical space measurements were used. The `P2L` function thus depends on the camera intrinsics as well as method-dependent measurements. The `EPSM` function does not play a role in the pose estimation process. In figure 3.7, the TU/e and HAN implementations are fiducial marker (FM) based. Other methods may or may not be FM based.

Device	Central Computer	Central Computer (1 thread)	Raspberry-Pi
Time	18.22	27.77	37.03

Table 3.1: Processing time for one frame.

3.4 Execution Time Analysis

Within the current scope, execution time is the interval between the moment when the first bit of data is sent by the sensors and the moment when the last bit has been received by the truck network interface. The current analysis focuses on impact of processing time, communication time and the scheduling of the two on overall execution time.

3.4.1 Processing Time

Processing time is understood as the time in the execution loop of a node necessary for data to be processed. The execution loop refers to the loop of operations that a node in the localization system performs. As shown in figure 3.8, the overall execution loop of the central computer in the centralized system can be split into 3 stages:

1. IEEE1394 Communication: input communication from cameras to central PC;
2. Pixel Data Processing: processing of pixel data to obtain global pose. This stage can involve pose information fusion for the CT implementation;
3. Image Show: displaying of images so user can see live camera feed with OpenCV axis drawn on markers.

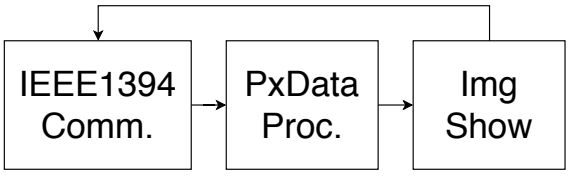


Figure 3.8: Centralized system loop

When analyzing the processing time, one must consider the hardware characteristics of the system processing nodes and, furthermore, how these are used. An individual fog node will, in most cases, have a larger processing time for one frame than the more powerful central computer. Table 3.1 shows the mean times for processing one frame for the central computer and Raspberry-Pi . The value under "Central Computer" shows the processing time when localization is done without any explicit control of threads. "Central Computer (1 thread)" shows the processing time when only one thread is assigned to process the frame.

With regard to how the processing node hardware resources are used, this thesis focuses on analysis of how processing time of the central computer can be reduced through parallelism. In both the

centralized and distributed systems, parallel processing of frame data has been implemented. For the centralized system, parallelism is achieved such that each camera is assigned one thread (CPU parallelism) to fetch the pixel data and process it. For the distributed system, there is one fog node assigned for each camera.

3.4.2 Speedup of centralized processing using threads

For the centralized system both sequential and threaded processing have been implemented. This allows theoretical estimation and measurement of actual speedup of the parallel vs. sequential processing. The speedup upper bound can be estimated using Amdahl's law [65]. The law is used when the workload is fixed and processing time is reduced by the parallelization. Amdahl's Law is formulated in the following way:

$$Speedup = \frac{1}{(1-p) + \frac{p}{s}} \tag{3.3}$$

where:

- *Speedup* is the overall speedup of the pixel data to global pose processing;
- s is the speedup of the code that can be parallelized;
- p is the percentage of the code that can be parallelized and benefiting from speedup s.

Threading has been applied on stages 1 and 2 of the central computer loop shown in figure 3.8. These stages have been measured to take 81.54% of the loop duration. Threading involves using 1 thread for each camera frame, therefore 8 threads in total. This does not however mean that the speedup s is equal to 8. Frame processing done by 1 thread is slower, as seen in table 3.1. The time for processing one frame without threading is 18.22 milliseconds, while the time for 1 thread is 27.77 milliseconds. The speedup s is therefore $= \frac{8 \cdot 18.22}{27.77} \approx 5.25$. Given these values, according to equation 3.3, the speedup is:

$$Speedup = \frac{1}{(1-0.8154) + \frac{0.8154}{5.25}} = 2.94 \tag{3.4}$$

The actual measured speedup of the processing loop is of 2.55. The comparison between sequential and threaded implementations can be found in table 3.2. The table shows the mean, jitter and standard deviation of the execution loop time measurements in milliseconds. One possible cause for the actual speedup being lower than theoretical is thread overhead. When using threads, the mean

Implementation	Loop mean	Jitter	Standard Deviation
Sequential	178.92	40.41	8.46
Threaded	70.28	40.33	6.04

Table 3.2: Execution loop time measurements

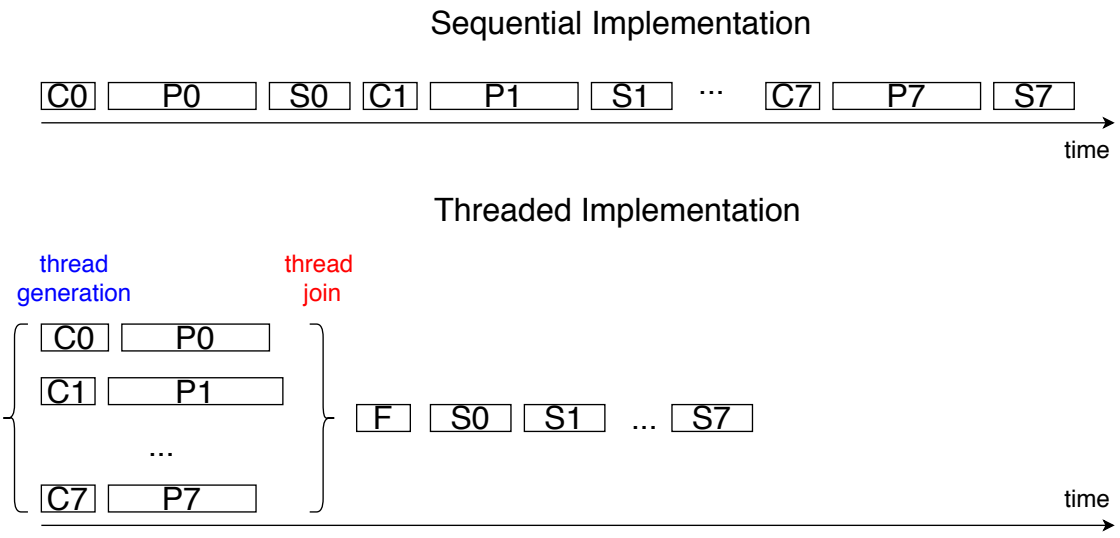


Figure 3.9: Sequential vs. threaded processing

processing time per frame increases from 18.22 to 27.77 milliseconds, so by 65.6%. Furthermore, occasional rises in IEEE1394 communication time were also noticed.

A visual comparison of sequential vs. threaded processing is shown in figure 3.9. `Cn` represents input communication time, `Pn` processing time and `Sn` display (show) time for camera frame `n`. The distributed implementation also performs fusion of pose information `F`.

3.4.3 Speedup achieved with fog nodes

For the distributed system, one parallel processing implementation has been realized. No parallel vs. sequential speedup estimation or measurement has been done. The scope is in comparing the speedup achieved from moving from a centralized to a distributed system. The performance comparison can be found in table 3.3. The table is an extension of table 3.2, with the last row added for the distributed implementation.

Comparing mean values, the distributed system performs 1.83 times faster than the centralized-sequential implementation. This is not as fast as the 2.55 speedup obtained by applying threads. The distributed implementation has a much higher loop time standard deviation than the centralized implementations. This is caused by the lack of system-wide synchronization of the Raspberry-

Implementation	Loop Mean	Jitter	Standard Deviation
Sequential	178.92	40.41	8.46
Threaded	70.28	40.33	6.04
Distributed	97.62	159.21	17.22

Table 3.3: Execution loop time measurements

Pi and central computer loops. The lack of synchronization causes high communication jitter which contributes to raising the mean loop time. FireWire with its isochronous cyclical transmission and Direct Memory Access (DMA) support can provide more deterministic and in this case faster communication time than Wi-Fi. Communication and communication-processing scheduling are discussed in the sections below.

3.4.4 Communication Time

Communication time is understood as the time in the execution loop necessary for communication of data to be done. This is not synonymous with transmission time, which is the time necessary for a network interface to transmit data on the network medium. Rather communication time is measured in-code, as the time spent by a node transmitting; receiving or waiting to receive necessary data. For the centralised system, it is the time necessary for phase 1 in figure 3.8 to be executed. For the decentralized computer it is the time spent by the central computer receiving or waiting for pose information from the fog nodes. In both systems, the central computer ultimately sends the pose information towards the trucks, however for simplicity this time is included in the processing phase.

3.4.5 Transmission time

The transmission time (T_t) between two network interfaces, assuming continuous transmission, can be determined using the following equation:

$$T_t = \frac{DataSize}{Bwidth} \tag{3.5}$$

where:

- $DataSize$ is the data size of the data to be sent and
- $Bwidth$ is the bandwidth (or data rate) allowed by the interface speed and the medium.

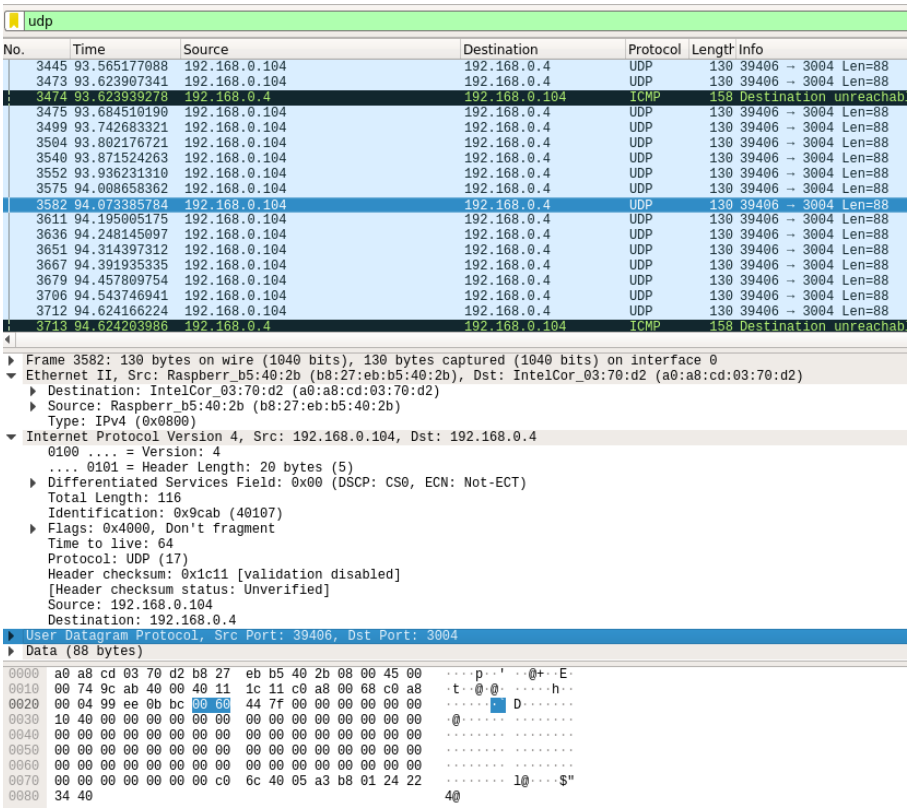


Figure 3.10: Capture of UDP datagrams sent by Raspberry Pi's

In the distributed system, the Raspberry Pi's transmit the pose data only, not the entire image data as this has already been processed. The data size is thus only 130 bytes as shown in figure 3.10. The frame size is shown in the layer 1 section of the Wireshark screenshot (above "Ethernet II"). The data is transmitted to the central computer using a Wi-Fi router with 150 Mbps interface. The transmission to/from router, according to equation 3.5 takes $\frac{130 \cdot 8}{150 \cdot 10^6} \approx 7$ microseconds. Not accounting for residence time in router, the best case communication time is thus 7 microseconds from Raspberry-Pi to router and 7 microseconds from router to central computer, thus 14 microseconds in total. The worst case is $14 + 5 \cdot 7 + 5 \cdot 7 = 84$ microseconds if the frame is queued behind the frames of the other 5 Raspberry Pi's.

In the case of the centralized system, the FireWire protocol is used with S400 speed (393.216 Mbps). The image size is 307200 bytes. If transmission time is calculated according to equation 3.5, then the result would be $\frac{307200 \cdot 8}{393.216 \cdot 10^6} = 6.25$ milliseconds. FireWire however uses isochronous transmission for determinism. It transmits in cycles that are split in multiple slots. Using the IEEE1394 library libdc1394 [66], the frame size and number of packets per frame were determined, as shown in figure 3.11. The protocol is set to send 480 packets per frame. FireWire has an average cycle time of $\approx$ 125 microseconds [44]. If each camera is assigned one isochronous frame per cycle, then the average transmission time for one frame is $125 \cdot 480 \cdot 10^{-6} = 0.06$ seconds or 60 milliseconds.

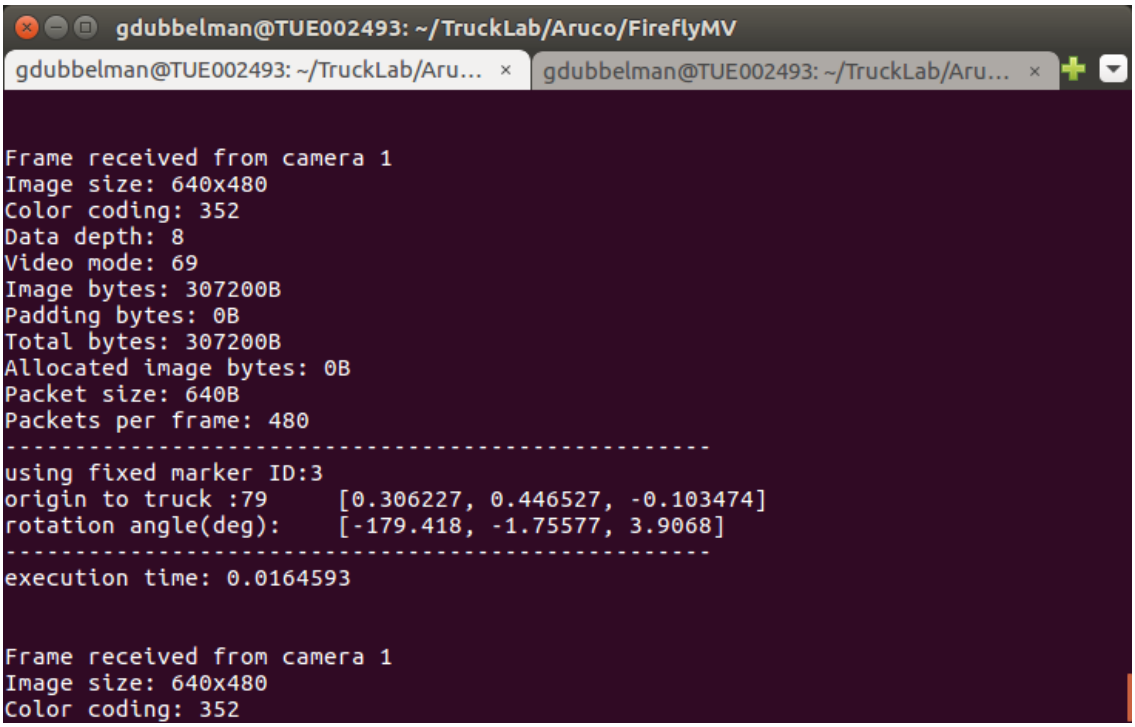


Figure 3.11: libdc1394 output describing FireWire behaviour

So far the transmission times point towards the centralized system having a slower communication that the decentralized system. In the next section it is explained why this is not the case and how communication-processing scheduling affects execution time.

3.4.6 Communication-Processing Scheduling

The synchronized and parallel execution of data transmission and processing tasks can have a significant effect on overall execution time. In the case of the centralized system, the increase in transmission time from continuous to isochronous is of $\frac{60}{6.25} = 9.6$. The protocol requires a higher transmission time in order to achieve deterministic cyclical transmission. The impact of transmission on execution time can however be minimized if parallelism is achieved with respect to processing tasks. In the case of FireWire , interfaces can be equipped with DMA chips [67]. If the FireWire interface uses DMA, then frame data can be transmitted to the adapter and written to main memory in parallel with CPU operations [44]. With DMA, a dedicated data transfer device reads incoming data from a device and stores that data in a system memory buffer for later retrieval by the processor. This DMA process occurs transparently from the processor's point of view [68].

3.4.7 Communication-Processing Parallelism in Centralized System

The current implementation is configured to use the FireWire interface DMAs. The code responsible for waiting for frame data to be received and returning the memory buffer of the frame has been measured to execute for only 0.0199 milliseconds, or approximately 20 microseconds. The worst-case measurement is of 0.0685 or 68.5 microseconds. This is much less than the computed average case time, which shows the speedup effect of using processing-communication parallelism. This indicates that that operations done by DMA and CPU are in parallel. In this case the CPU only polls the DMA and until DMA has finished writing frame data to memory, the CPU can perform its own operations.

To further understand the DMA behaviour, tests were run where only the code responsible for communication was run within the execution loop. For the sequential implementation, instead of the usual `C0 -> P0 -> S0 -> C1 -> P1 -> S1 -> ...`, the loop was reduced to solely communication: `C0 -> C1 -> C2 -> ...`, similarly for parallel. The tests revealed the behaviour shown in figure 3.12. The communication times for the first 7 camera frames were much shorter than the last. Why the communication with the last camera is always longest in the threaded implementation is unknown, although this may be determined by the order in which the threads are generated. Table 3.4 shows the mean communication time for both implementations for each cam number. The last column shows the jitter, which is much lower than for best-effort communication due to the FireWire isochronous behaviour.

Impl./Cam no.	0-6 (mean)	7 (mean)	7 (jitter)
Sequential	0.028	65.610	0.678
Threaded	0.027	66.130	0.448

Table 3.4: FireWire with DMA behaviour

The measured times indicate to the the possibility that, for the first frames, the CPU is only polling. For the last frame, it is possible that the CPU has to wait before polling for the first frame again. To understand the process in full, further analysis is required. The low measured communication times for the execution loops do indeed indicate that the CPU polls for the DMA to start the transfer of image data to memory while CPU performs tasks on that data, as shown in figure 3.13. The figure shows what DMA and CPU behaviour could look like for both the actual implementation and communication-only behaviour.

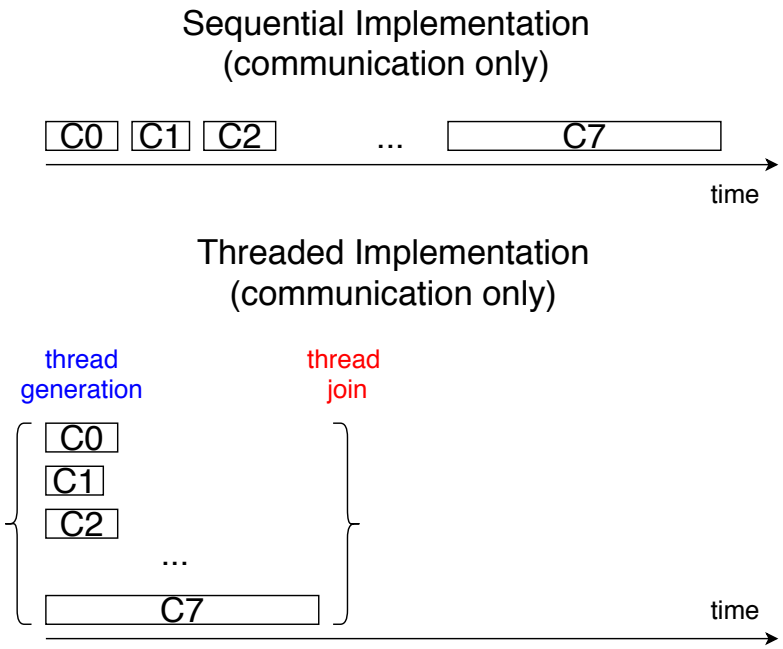


Figure 3.12: FireWire with DMA behaviour

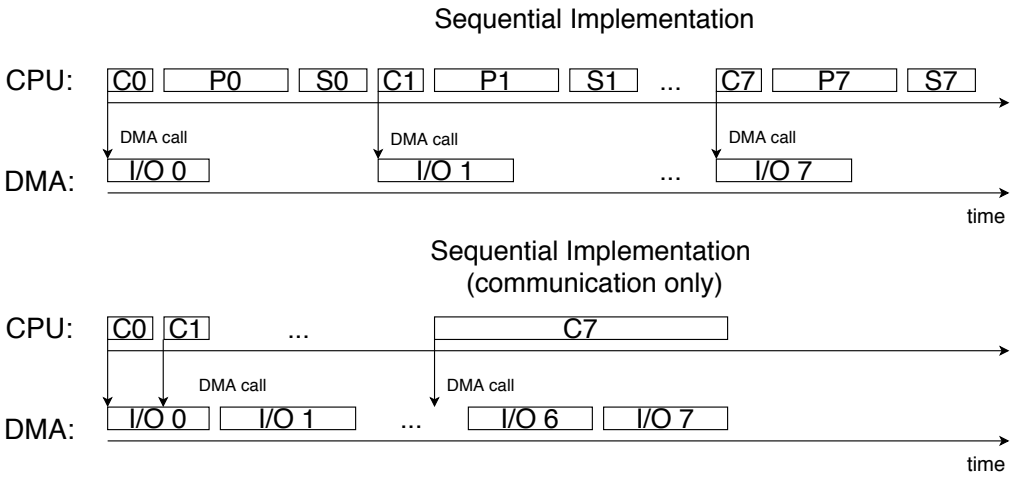


Figure 3.13: DMA calls

3.4.8 Synchronization Problem in Decentralized System

The lack of loop synchronization among the distributed system devices can lead to large waiting times. As shown in table 3.3, on average the distributed system loop is measured to take 97.62 milliseconds. The standard deviation of 17.22 and jitter of 159.21 milliseconds are however much higher than the values for the centralized implementations. Separate measurements were done to determine the communication time spent within a loop. They show that most loop time is spent during communication, with an average of 63.23 and a jitter of 154.31 milliseconds. As in the centralized system situation, the impact of the communication time is not reflected by the transmission time. For the centralized system the communication time was lower than the transmission time, however this time it is much higher. This is due to lack of synchronization between the central computer and Raspberry-Pi loops. The decentralized system assigns one thread for each Raspberry-Pi such that each thread is responsible for receiving a UDP frame from its associated Pi. The CPU must wait until all threads are done receiving the UDP frames from the Pi's. Thus the communication time will always be pushed up by the Pi that sends last. A theoretical unsynchronized scheduling example is shown in figure 3.14, which is similar to the real system behavior concerning jitter. The Raspberry-Pi schedules are shown in the `RPn` rows, where `n` is the Raspberry-Pi number. The UDP frames that are received and sued by the central computer are shown in blue, the dropped frames are shown in red. In this example Raspberry-Pi no. 4 sent the pose information just a bit sooner than the start of the communication time of the computer. This leads to its UDP frame being lost because the central computer is not listening for any frames. Instead it must send the pose information in the next loop. The other frames are dropped because the Raspberry Pi's send either:

- during central computer communication time, when their associated threads have already passed communication phase;

- during processing (P) or frame show (S) time, when the CPU is performing non-communication tasks.

Note that in this example the central computer spends most communication time waiting rather than receiving frames. Figure 3.15 on the other hand shows a synchronized schedule such that communication time is reduced and waiting time is minimized. In this case all frames are received and used. In the following chapter, proposals for achieving such a synchronized scheduling are discussed.

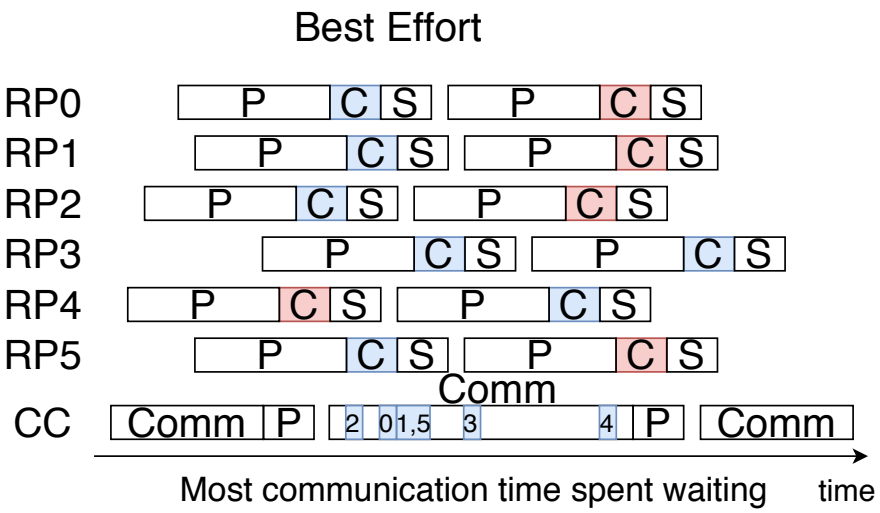


Figure 3.14: Unsynchronized best effort scheduling

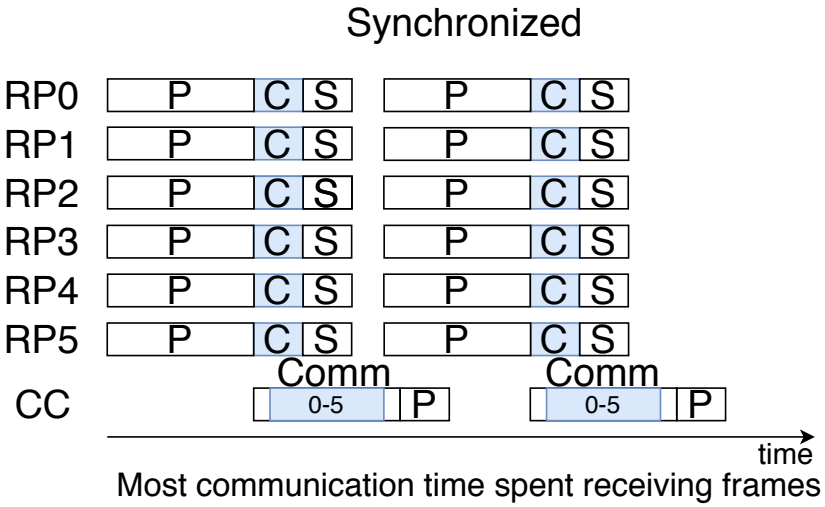


Figure 3.15: Synchronized scheduling

3.4.9 Data Reduction

It should be noted that data reduction occurs as a result of fog node processing. The Raspberry Pi's process the image pixel data of 921600 bytes into pose data of 130 bytes. The data reduction ratio is thus of 7809.23. This of course impacts the theoretical transmission time, reducing it with the same ratio for the same bandwidth. In systems that are synchronized and where transmission time has a heavier impact on execution time, such data reduction could reduce the execution leading to better real time performance. This of course comes at the cost of data loss or the need of implementing storage or transmission of data to separate hardware.

Chapter 4

Discussion and Conclusion

Based on the experiments and analysis performed the following answers can be provided for the research questions:

1. How is the truck pose estimated by the TU/e Trucklab localization system and in general?

   The localization process consists of the following steps:

   - Detect the markers within a frame.
   - Extract translation and rotation vectors.
   - Because multiple cameras are used, extra calculations are performed to transition from local reference (moving to fixed marker) to global reference (moving to `E0`).

   To generalize, when localizing, the points of interest must first be identified within the frame. Second, these points must be used to derive localization information using physical units. In our case, translation and rotation vectors are obtained which help derive the marker global pose. Another case could be localizing the truck corners and deriving the truck pose based on known distances between the corners.

2. What are the system parameters that influence accuracy?

   The localization process tells us that we need to account for camera intrinsics and provide accurate measurements. In the case of OpenCV the camera intrinsics are the camera matrix and the distortion coefficients. Accurate measurements must be done because these are inputs for the pixel-to-physical-unit conversion. In our case, there is the marker length and, more importantly, the distance from fixed marker to global reference origin (`E0`). The measurement between fixed marker and origin are part of the equations used for local to

global transition. These are $\texttt{tvec}_{\texttt{E0}}$ and $\mathcal{R}_{\texttt{E0}}$ used in equation 3.2. If the truck corners were to be used, then accurate measurements of the inter-corner distances would have to be made.

3. What are the system parameters that influence execution time?

   Execution time is the sum of the time spent by the system to process and communicate data. Time spent processing, if done sequentially, can be improved by applying parallelism. For the centralized system, CPU parallelism has been employed. GPU parallelism could perhaps provide better performance, however implementing it is left for future work. Time spent communicating can be reduced by performing it without having the CPU to stop and wait. This is an important difference between the centralized and decentralized systems. While transmission time for centralized is theoretically higher, the time spent by the CPU with communication operations is much less on average than for the distributed system. This is due to to DMA use which allows for processing-communication parallelism. The distributed system does not perform optimally concerning communication as large waiting times can occur due to frame information being transmitted without any kind of fog node and central computer synchronization. For the decentralized system, if transmission time heavily influences communication time, then transmission time can be minimized by increasing the link bandwidth or by reducing the transmitted data size. Data reduction can be achieved with the trade-offs of discarding data or storing it in memory and transmitting only the essential data.

4. How will a decentralized system perform in comparison to the current centralized system?

   The decentralized system without synchronization is faster than the centralized sequential system but slower than the centralized threaded system. Further implementation work must be done in order to measure the performance of a synchronized distributed system. Given the time it takes for a Raspberry-Pi to process a frame ($\approx$ 37 milliseconds), synchronization might lead to faster execution time than the centralized threaded system. The centralized system itself can also be further improved by implementing GPU parallelism. In terms of scalability, the decentralized system allows for an indefinite number of devices in theory. FireWire allows for a maximum 63 devices on the bus. If a large scale system is to be used, bus connections could limit the number of devices used.

Given the above, this chapter will further discuss improvement proposals for the decentralized system.

4.1 Proposed Improvements for the Decentralized System

Standard IT network equipment such as the one used in the distributed implementation has no concept of "time" and cannot provide synchronization, precise transmission timing and upper bound for delays [69]. On average, real-time constraints may be satisfied however the jitter caused by lack of synchronization may be unacceptable, especially for safety-critical applications. In the current context, large delays may lead to truck collisions.

4.1.1 Synchronization

For network devices to operate in unison and execute their operations at the required points in time, clock synchronization is necessary. This can be achieved with GPS clocks however there is no guarantee that small devices such as fog nodes have access to radio or satellite signals. The 802.1AS standard [70] describes generalized Precision Time Protocol (gPTP). The protocol can be used to establish a common time reference between the central computer and the fog nodes, or between the fog nodes if the latter is absent.

In an isolated network, where no other traffic is present except localization system traffic, clock synchronization should be enough to minimize waiting time. Fog nodes can be set to transmit at specified time intervals and thus the processing node that receives it will have a bounded waiting time. In figure 4.1, a sample use case is shown, where fog nodes with synchronized clocks send frames at known time intervals. The scheduling is done such that the frames arrive in constant order $1, 2, 3, 4$, however this is not always required. The router in the network does not require to have its clock synchronized as the periodic frame arrival is ensured by the periodic transmission by the fog nodes. For the network shown in figure 4.1, the frames received by the central computer are shown in figure 4.2.

4.1.2 Integration with Separate Network

If the localization is to be integrated as part of a larger distribution center network, then the data transmitted by the fog nodes may have to share the network with other traffic. This can lead to large delays due to overcrowding or even packet loss due to buffer overflows. In order to ensure timely arrival of localization system frames, the traffic shaping mechanisms described in Time-Sensitive Networking (TSN) 802.1Qav [71] and 802.1Qbv [72] standards can be implemented.

802.1Qav is the standard for credit-based traffic shaping, which ensures bounded latency for two traffic priority classes. One such maximum latency is computed in the 802.1BA standard, which specifies LAN component standards. The bounded latencies for the two classes (A and B) are

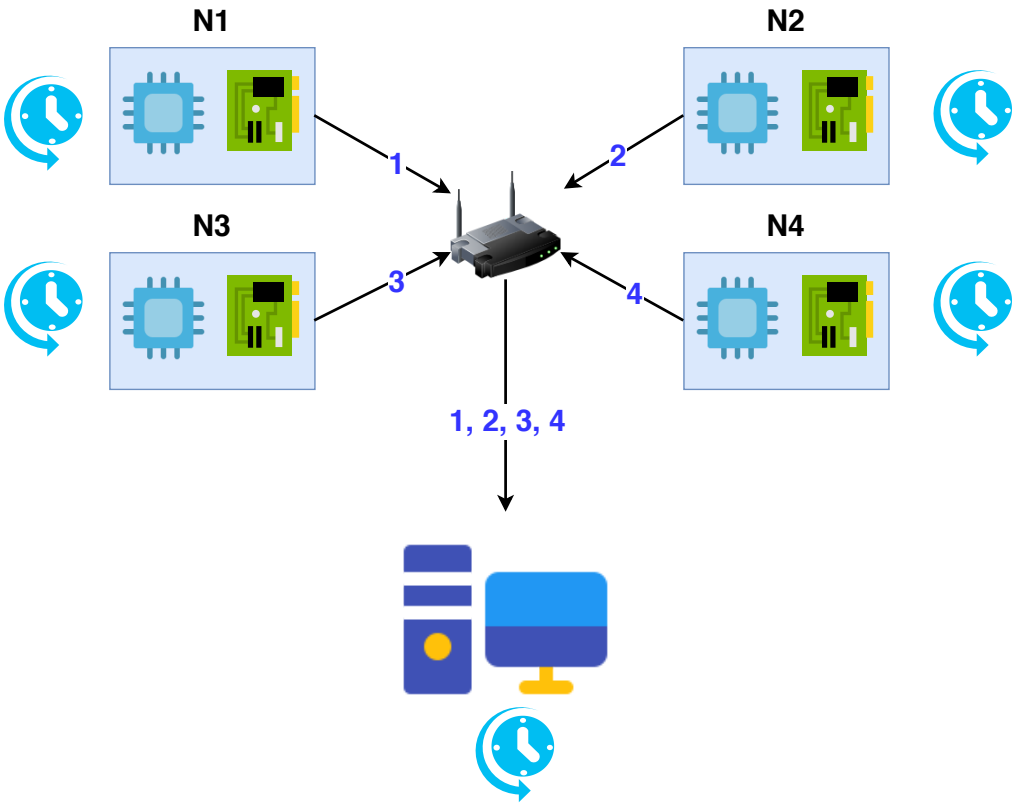


Figure 4.1: Sample use case with 802.1AS clock synchronization

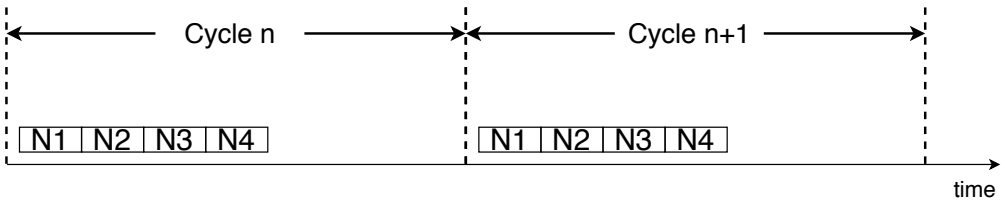


Figure 4.2: Periodic frame receival for central computer in figure 4.1

Stream Class	Max. end-to-end delay
A	2 ms
B	50 ms

Table 4.1: Latencies specified by 802.1BA for network employing credit-based traffic shaping

shown in table 4.1. In this case they are computed for a network using at least 100Mbps wired Ethernet and for a maximum of 7 hops. Implementing credit-based traffic allows for assigning more bandwidth to Audio Video (AV) streams and preventing frame loss thus preserving QoS. The prevention of frame loss is achieved by evening out the traffic and reducing bursting and bunching which can lead to buffer overflows. At the same time the 75% limit for AV traffic protects the best-effort frames of separate network.

802.Qbv specifies time-aware scheduling of traffic, which allows shaping of communication into fixed-length, repeating time cycles, similar to TDMA. Within such a cycle, exact transmission slots can be given to each type of traffic on the network. This allows for time critical traffic to be separated from non-critical background traffic. Figure 4.3 showcases an example network where pose information (critical) along with other (non-critical) streams are present. The scheduling of the router is now formed using the time-aware shaper, as shown in figure 4.4. The time required for pose information transmission is allocated in a similar way as the isochronous traffic in FireWire . This provides highly deterministic behaviour in the presence of non-critical background traffic.

To conclude, if the localization system is to be used in the presence of other network traffic:

- 802.1Qav is more suited if entire video streams from cameras are to be transmitted towards processing devices for pose extraction or live streaming.
- If only pose information is to be communicated, 802.1Qbv can be used to allow hard real-time communication delay to small scaled pose information. 802.1Qbv can also be used for video streams however this may cause large delays for other traffic. Time-aware shaper can be useful if separate streams are on the network, so as to provide pre-determined transmission windows for pose information traffic.

Given the above, for the current distributed implementation, 802.1AS may be sufficient for the Raspberry Pi's to transmit with periodic behaviour and in synch with the central computer.

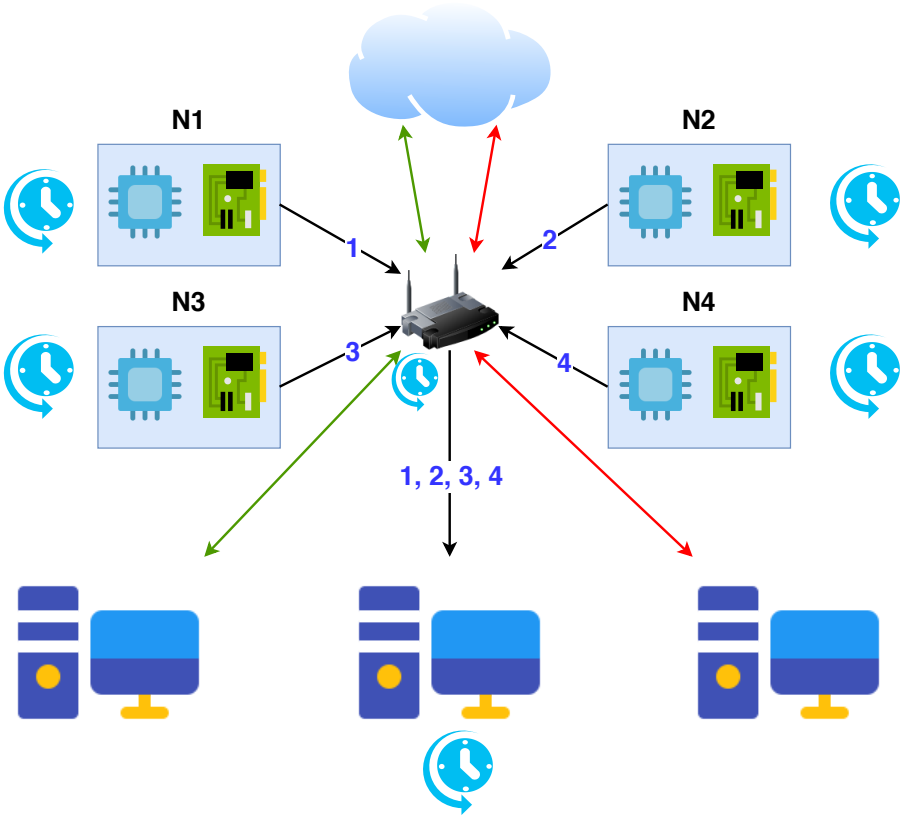


Figure 4.3: Sample use case with 802.1Qbv implementation

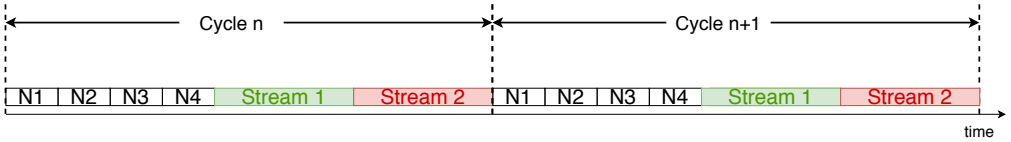


Figure 4.4: Fixed-length transmission time schedule on router in figure 4.3

4.2 Conclusion and Future Work

As part of this work, the localization process of the TU/e truck localization system has been explained. Based on the particular localization processes used in TU/e and in HAN, the general process can be described. To summarize, the general process consists of point-of-interest identification followed by pose estimation and optionally local to global pose transitioning. Parameters concerning accuracy and execution time have been identified. Accuracy is influenced by the camera intrinsic parameters and by the correctness of point of interest measurements. Execution time is influenced by processing and communication times, as well as the scheduling between the two. Based on the experimental conclusions, improvements have been proposed to optimize the execution time of the distributed system. The improvement consists of employing clock synchronization for the fog nodes and central computer so that periodic behaviour and synchronized transmission can be ensured. Below, future work is proposed, divided into topics:

- Accuracy:
  - In-depth analysis of camera intrinsic parameters and their effect on accuracy.
- Execution time:
  - Analysis of other hardware aspects affecting frame processing time such as memory speed.
  - Exploration of speedup obtainable by other parallel processing methods such as GPU.
  - Implementation of speeded up ArUco detection, as proposed in [73].
  - Implementation of synchronized periodic scheduling for the distributed system.

Bibliography

[1] Wikipedia. Logistics, 2019. URL: https://en.wikipedia.org/wiki/Logistics. 2

[2] Christopher Morris. *Academic Press Dictionary of Science and Technology*. Academic Press, Cambridge MA, 1 edition, 1992. 2

[3] US Bureau of Transportation Statistics. Weight of shipments by transportation mode. Data used for chart can be found at: https://www.bts.gov/weight-shipments-mode, 2019. 2, 3

[4] Eurostat. Modal split in the eu, 2019. URL: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Freight_transport_statistics_-_modal_split. 2, 3

[5] Eurostat. Road transport by maximum permissible laden weight and load capacity of vehicle, 2019. URL: https://ec.europa.eu/eurostat/statistics-explained/index.php/Road_freight_transport_by_vehicle_characteristics (statistics year has changed since time of writing). 2

[6] Wikipedia. Logistics automation. URL: https://en.wikipedia.org/wiki/Logistics_automation, 2019. 2

[7] DHL. Robotics in logistics a dpdhl perspective on implications and use cases for the logistics industry, 2016. 4

[8] J. Visser. Automated Freight Transport. In *1st Chinas Symposium on Underground Freight Transportation - Shanghai 2017*, 2017. 4

[9] G. Rizzoni. Transformational Technologies Reshaping Transportation – An Academia Perspective. In *SAE 2019 Commercial Vehicle Engineering Congress*, 2014. 5

[10] Society of Automotive Engineers (SAE) International. Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles (J3016). https://saemobilus.sae.org/content/j3016_201806, 2018. 5

[11] Daimler. Automated driving at daimler trucks, 2020. URL: https://www.daimler.com/innovation/case/autonomous/automated-driving-daimler-trucks.html. 5

[12] DAF. DAF EcoTwin truck platooning. https://www.daf.com/en/about-daf/innovation/daf-ecotwin, 2020. 5

[13] Ensemble. The Project. http://www.platooningensemble.eu/project, 2020. 5

[14] Society of Automotive Engineers (SAE) International. SAE Standards News: J3016 automated-driving graphic update. https://saemobilus.sae.org/content/j3016_201806, 2019. 6

[15] Y. Ding et al. Vehicle Aided Positioning Method Based on Intelligent Identification. In *2019 34rd Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, 2019. 6

[16] Wikipedia. Inertial navigation system. URL: https://en.wikipedia.org/wiki/Inertial_navigation_system, 2020. 6

[17] N. Ganganath et al. MOBILE ROBOT LOCALIZATION USING ODOMETRY AND KINECT SENSOR. In *Emerging Signal Processing Applications (ESPA), 2012 IEEE International Conference on, At Las Vegas, USA*, 2012. 6

[18] D. A. Nguyen and K. H. Minh. A research on locating agv via rss signals. March 2019. 6

[19] A. Azenha. A Neural Network Approach for AGV Localization Using Trilateration. 2009. 7

[20] C.. Roerig et al. Localization of Autonomous Mobile Robots in a Cellular Transport System. In *Engineering Letters, 20:2, EL_20_2_05*, 2012. 7

[21] C. Roerig et al. Localization of an Omnidirectional Transport Robot Using IEEE 802.15.4a Ranging and Laser Range Finder. In *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010)*, 2010. 7

[22] C. Kirsch et al. Global Localization and Position Tracking of an Automated Guided Vehicle. In *18th World Congress The International Federation of Automatic Control Milano (Italy) August 28 - September 2, 2011*, 2011. 7

[23] C. Roerig et al. Global Localization and Position Tracking of Automatic Guided Vehicles using passive RFID Technology. In *Proceedings of the joint 45th International Symposium on Robotics (ISR)*, 2014. 7

[24] R. Kusumakar. Autonomous Parking for Articulated Vehicles. Master's thesis, Hogeschool Arnhem en Nijmegen (HAN), Netherlands, 2017. 7, 8, 26

[25] A. Lucas et al. Mosaic Based Flexible Navigation for AGVs. 2010. 7

[26] P. Yuan et al. AGV System Based on Multi-sensor Information Fusion. In *2014 International Symposium on Computer, Consumer and Control*, 2014. 7

[27] L. Juntao et al. Research of AGV Positioning Based on the Two-Dimensional Code Recognition Method. 2015. 7

[28] M. Badshah, I. Automated Dockingofascaled TractorSemi-Trailer. Master's thesis, Technische Universiteit Eindhoven, Netherlands, 2018. 8, 55, 56

[29] A. Sudhakaran. Autonomous Parallel Parking ofa Scaled Tractor Semi-Trailer. Master's thesis, Technische Universiteit Eindhoven, Netherlands, 2018. 8

[30] A. M. Hertogh, M. Development of a virtual environment and supervisory control for Trucklba. Master's thesis, Technische Universiteit Eindhoven, Netherlands, 2018. 8

[31] T. Hertogh. Automatic docking of articulated vehicles, 2017. 8

[32] S. Rajagopalan. MATLAB basedControl ofa Scaled TractorSemi-trailer, 2017. 8

[33] R. Oma et al. A Tree-Based Model of Energy-Efficient Fog Computing Systems in IoT. In *Complex, Intelligent, and Software Intensive Systems*, 2018. 8

[34] R. Oma et al. An Energy-efficient Model of Fog and Device Nodes in IoT. In *32nd International Conference on Advanced Information Networking and Applications Workshops*, 2018. 8

[35] B Negash et. al. Fog Computing Fundamentals in the Internet-of-Things. In *Fog Computing in the Internet of Things*, 2018. 10, 11

[36] Rudy Montgelas (Otronics). ENABLING TOMORROW'S CONNECTED INFRASTRUCTURE FOR IoT. Technical report, 2016. 10, 12

[37] IoT Agenda. internet of things (iot). URL: `https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT`. 10

[38] Alcatel Lucent. Enabling the Internet of Things and Digital Business. Technical report, 2017. 11, 12

[39] J. (Deloitte) Holdowsky et al. Inside the Internet of Things (IoT). Technical report, 2016. 11, 12

[40] Eric Olson. What is the most popular industrial network protocol? URL: https://insights.globalspec.com/article/11936/what-is-the-most-popular-industrial-network-protocol, 2019. 12

[41] Cisco. IEEE 802.11ax: The Sixth Generation of Wi-Fi. Technical report, June 2018. 12, 13

[42] IEEE Computer Society. IEEE Standard for a High Performance Serial Bus. December 1995. 12

[43] IEEE Computer Society. IEEE Standard for a High-Performance Serial Bus—Amendment 2. December 2002. 12

[44] D. Staikos. Firewire reference tutorial (an informational guide), 2010. URL: http://www.1394ta.org/press/whitepapers/firewire%20reference%20tutorial.pdf. 12, 31, 32

[45] WG802.3 Ethernet Working Group. IEEE 802.3an-2006 - IEEE Standard for Information Technology - Telecommunications and Information Exchange Between Systems – LAN/MAN - Specific Requirements Part 3: CSMA/CD Access Method and Physical Layer Specifications - Amendment: Physical Layer and Management Parameters for 10 Gb/s Operation, Type 10GBASE-T. June 2006. 12

[46] WG802.3 Ethernet Working Group. IEEE 802.3ba-2010 - IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 3: CSMA/CD Access Method and Physical Layer Specifications Amendment 4: Media Access Control Parameters, Physical Layers, and Management Parameters for 40 Gb/s and 100 Gb/s Operation. June 2010. 12

[47] E Spurgeon et al, C. *Ethernet The Definitive Guide*. O'Reilly, 2014. 12

[48] P. Krejov et al. Intel RealSense Depth Camera over Ethernet. 12

[49] Wikipedia. Wi-fi. URL: https://en.wikipedia.org/wiki/Wi-Fi, 2020. 13

[50] Dennis Huang (Ruckus). Wi-fi 6 fundamentals: What is 1024-qam? URL: https://theruckusroom.ruckuswireless.com/wired-wireless/technologytrends/wi-fi-6-fundamentals-1024-qam/, 2018. 13

[51] Qorvo. Wi-fi 6 (802.11ax): 5 things to know. URL: https://www.qorvo.com/design-hub/blog/80211ax-5-things-to-know, 2017. 13

[52] S. Garrido-Jurado et al. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280 – 2292, 2014. 13, 15

[53] OpenCV. Detection of aruco markers. URL: https://docs.opencv.org/master/d5/dae/tutorial_aruco_detection.html, 2019. 14, 21

[54] M. Fiala. Designing Highly Reliable Fiducial Markers. In *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 32, NO. 7*, 2010. 15

[55] A. Sagitov et al. Comparing Fiducial Marker Systems in the Presence of Occlusion. In *International Conference on Mechanical, System and Control Engineering*, 2017. 15

[56] Opencv - about. https://opencv.org/about/. Accessed: 2019-10-01. 15

[57] S. Kusumakar, S. Tomar, A., K. Kural, and B. Pyman. Autonomous Parking for Articulated Vehicles. Master's thesis, Hogeschool van Arnhem en Nijmegen, 2017. 16

[58] Robotis. Turtlebot3 specifications. URL: http://emanual.robotis.com/docs/en/platform/turtlebot3/specifications, 2020. 16

[59] Raspberry pi 3: Specs, benchmarks & testing. https://www.raspberrypi.org/magpi/raspberry-pi-3-specs-benchmarks/. Accessed: 2019-10-01. 17

[60] raspberrypi.org. Camera module, 2018. URL: https://www.raspberrypi.org/documentation/hardware/camera/README.md. 17

[61] OpenCV. Aruco marker detection. URL: https://docs.opencv.org/4.2.0/d9/d6a/group__aruco.html, 2019. 21, 25

[62] OpenCV. Camera calibration and 3d reconstruction. URL: https://docs.opencv.org/master/d9/d0c/group__calib3d.html, 2019. 21, 25, 26

[63] R. M. Salinas. Aruco: An efficient library for detection of planar markers and camera pose estimation. URL: https://docs.google.com/document/d/1QU9KoBtjSM2kF6ITOjQ76xqL7H0TEtXriJX5kwi9Kgc, 2019. 21

[64] Salinas R. M. Aruco reference systems and the role of calibration. URL: https://youtu.be/k8ZSzofAKTk, 2018. 21, 22

[65] Wikipedia. Amdahl's law. URL: https://en.wikipedia.org/wiki/Amdahl%27s_law, 2020. 28

[66] D. Ducamp. libdc1394 homepage, 2020. URL: https://damien.douxchamps.net/ieee1394/libdc1394. 31

[67] Wikipedia. Direct memory access. URL: https://en.wikipedia.org/wiki/Direct_memory_access, 2020. 32

[68] F. Harvey et al. (National Instruments), A. Dma fundamentals on various pc platforms, 1991. 32

[69] Wikipedia. Time-sensitive networking. URL: https://en.wikipedia.org/wiki/Time-Sensitive_Networking. 40

[70] IEEE. 802.1AS-2011 - IEEE Standard for Local and Metropolitan Area Networks - Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks. 2011. https://ieeexplore.ieee.org/servlet/opac?punumber=5741896. 40

[71] IEEE. 802.1Qav-2009 - IEEE Standard for Local and metropolitan area networks– Virtual Bridged Local Area Networks Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams. 2009. https://ieeexplore.ieee.org/document/8684664. 40

[72] IEEE. 802.1Qbv-2015 - IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic. 2015. https://ieeexplore.ieee.org/document/8613095. 40

[73] F. Romero-Ramirez et. al. Speeded Up Detection of Squared Fiducial Markers. In *Image and Vision Computing*, 2018. 44

[74] Wikipedia. Raspberry pi. https://en.wikipedia.org/wiki/Raspberry_Pi. Accessed: 2019-10-01. 53

[75] Raspberry pi 3 model b launches today - 64-bit quad a53 1.2 ghz bcm2837. https://raspi.tv/2016/raspberry-pi-3-model-b-launches-today-64-bit-quad-a53-1-2-ghz-bcm2837. Accessed: 2019-10-01. 53

Appendix A

FFMV-03M2MC Specifications

Imaging Sensor	Micron 1/3" Wide-VGA CMOS
	MT9V022177ATM (BW) MT9V022177ATC (Color)
Shutter Type	Global shutter using Micron TrueSNAP™ technology
Active Imager Size	4.51mm (H) x 2.88m (V), Diagonal 5.35mm (1/3" type)
Active Pixels	752(H) x 480(V)
Pixel Size	6µm(H) x 6µm(V)
A/D Converter	On-chip 10-bit analog-to-digital converter
Video Data Output	8 and 16-bit digital data (see *Supported Data Formats* below)
Standard Resolutions	640x480
Frame Rates[1]	60, 30, 15, 7.5 FPS
Partial Image Modes	Pixel binning and region of interest modes available via Format_7
Interfaces	6-pin IEEE-1394a for camera control, video data transmission and power
	7-pin JST GPIO connector, 4 pins for trigger and strobe, 1 pin +3.3 V, 1 V_{EXT} pin for external power
Voltage Requirements	8-32V via IEEE-1394 cable or GPIO connector (V_{EXT})
Power Consumption	Less than 1W
Gain	Automatic/Manual Gain modes
	0dB to 12dB
Shutter	Automatic/Manual Shutter modes
	0.03 ms to 512 ms (extended shutter mode)
Gamma	0 to 1 (enables 12-bit to 10-bit companding)
Trigger Modes	IIDC v1.31 Trigger Modes 0 and 3
Signal To Noise Ratio	52 dB
Dimensions	44 mm x 34 mm x 24.38 mm (case enclosed)
Mass	37 grams (including tripod adapter)
Camera Specification	IIDC 1394-based Digital Camera Specification v1.31
Emissions Compliance	Complies with CE rules and Part 15 Class B of FCC Rules.
Operating Temperature	Commercial grade electronics rated from 0° - 45°C
Storage Temperature	-30° - 60°C
Operating Relative Humidity	20 to 80% (no condensation)
Storage Relative Humidity	20 to 95% (no condensation)

Appendix B

Raspberry Pi 3 Model B Specifications

The Raspberry-Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote teaching of basic computer science in schools and in developing countries. The original model became far more popular than anticipated, selling outside its target market for uses such as robotics [74].

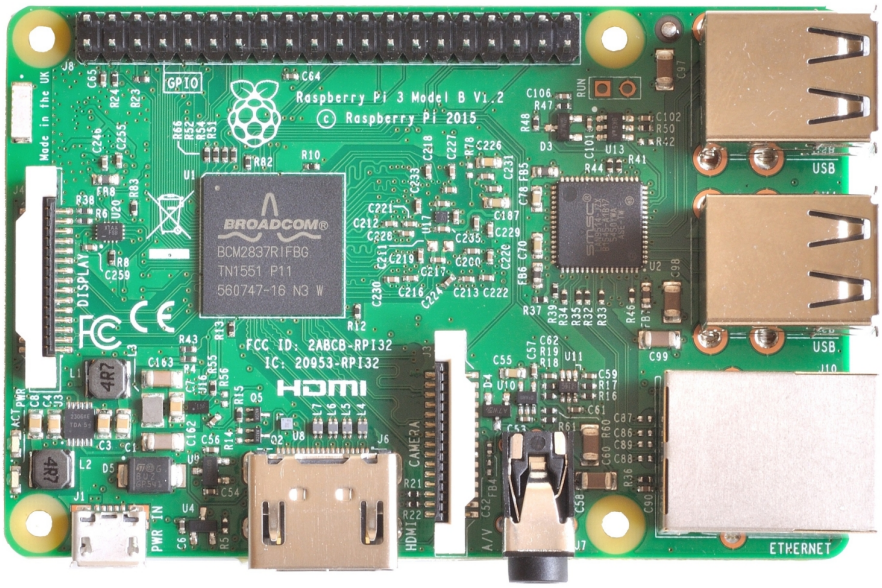


Figure B.1: Raspberry Pi 3 Model B [75]

- SoC: Broadcom BCM2837

- CPU: 4× ARM Cortex-A53, 1.2GHz
- GPU: Broadcom VideoCore IV
- RAM: 1GB LPDDR2 (900 MHz)
- Networking: 10/100 Ethernet, 2.4GHz 802.11n wireless
- Bluetooth: Bluetooth 4.1 Classic, Bluetooth Low Energy
- Storage: microSD
- GPIO: 40-pin header, populated
- Ports: HDMI, 3.5mm analogue audio-video jack, 4× USB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)

Appendix C

Truck Lab Dimensions

Figure shows the DC area dimensions as reported in [28]. Its scaling ratio is 1:13.3, with a length of 7.73 m and width of 4.87 m. There are a total of 10 docking bays and 3 parallel parking lots. Each docking bay loading station has a width of 0.28 m. The parallel parking dimensions 1.33 x 0.27 m. There is a free space of 1.22 m between a docked truck and a parked truck. The there are an 0.9 m wide entrance and an exit on both sides of the DC area.

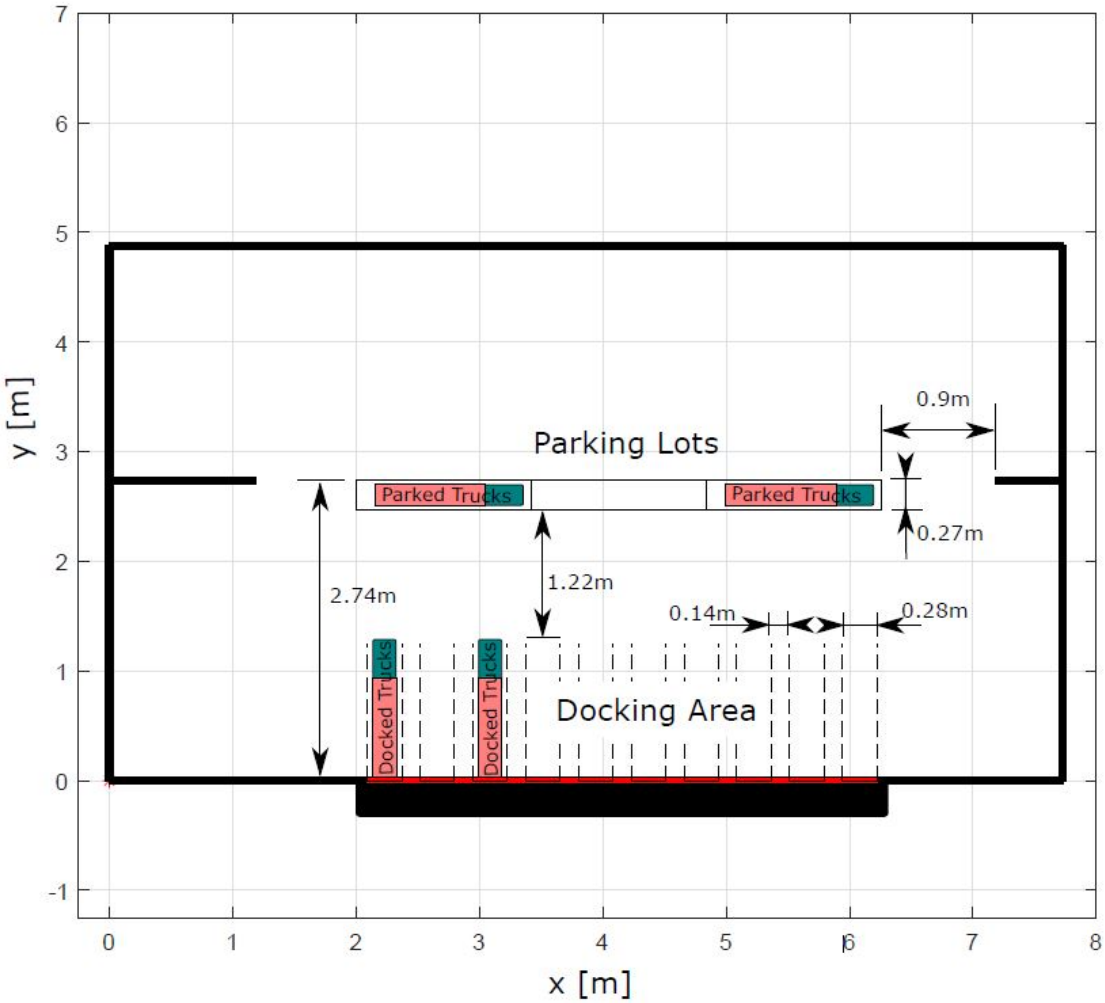


Figure C.1: DC area dimensions [28]