Zurich University
of Applied Sciences

School of
Engineering

IMS Institute of
Mechatronic Systems

Project work (BS Electrical Engineering)

Real-time communication using 5G to enable deep learning on the cloud for robotic applications

Author	Rafael Monteiro Marques
Main supervisor	Prof. Dr. Hans Wernher van de Venn
Sub supervisor	Charith Munasinghe
Date	24.05.2024

Declaration of originality

I hereby declare that I have written this thesis independently or together with the listed group members.

I have only used the sources and aids (including websites and generative AI tools) specified in the text or appendix. I am responsible for the quality of the text and the selection of all content and have ensured that information and arguments are substantiated or supported by appropriate scientific sources. Generative AI tools have been summarized by name and purpose.

Any misconduct will be dealt with according to paragraphs 39 and 40 of the General Academic Regulations for Bachelor's and Master's Degree courses at the Zurich University of Applied Sciences (Rahmenprüfungsordnung ZHAW (RPO)) and subject to the provisions for disciplinary action stipulated in the University regulations.

[illegible]

Winterthur, 24.05.2024 Rafael Monteiro Marques

Abstract

This thesis explores the use of advanced communication protocols to enable real-time communication for deep learning applications in robotics, specifically targeting crack detection. Despite initial plans to integrate 5G technology, the project proceeded with Wi-Fi-based solutions due to unforeseen challenges. The project involved developing a robust machine learning model for crack detection, assembling a portable inspection device with a camera and a portable computer, and establishing reliable communication channels for real-time data transmission and decision-making. The research validated off-board ML processing over Wi-Fi, highlighting the system's performance and areas for improvement. The study concludes with a comparative analysis of network setups, demonstrating significant insights into the potential and limitations of Wi-Fi in real-time robotic applications.

Keywords: Real-time communication, deep learning, robotics, crack detection, machine learning model, Wi-Fi, off-board processing.

Table of Figures

Table of Tables

Table of Contents

1. Introduction

1.1. Background

1.1.1. Current Capabilities of 5G

Fifth generation (5G) telecommunication technology represents a significant leap forward in digital communications, extending far beyond mere enhancements in mobile phone connectivity [1]. Characterised by its robust data transfer capacity, 5G facilitates high-speed internet capable of managing streaming ultra-high-definition videos and sophisticated virtual reality applications, thereby democratising modern digital experiences. This surge in performance is pivotal for the proliferation of applications such as remote surgeries and other health-related uses, where real-time video transmission requires unparalleled stability. `

Moreover, 5G substantially improves communications across a wide range of devices within the Internet of Things (IoT), enhancing these interactions to be more efficient and less energy consuming [2]. It also lays the foundational infrastructure for applications that demand immediate response times and high reliability, crucial for automated transportation systems and deploying machine learning models for real-time analysis in robotics.

As the global adoption of 5G-powered applications edges closer, evidenced by the European Union's ambitious agenda to secure comprehensive 5G coverage across all populated areas by 2030, the enhancements promise not only speed but also support for new types of services that are more reliable and widely accessible [3]. These advancements are instrumental in setting the stage for future innovations, particularly in fields requiring the integration of AI and real-time data processing, like robotic applications.

1.1.2. Challenges of 5G: Latency in Critical Applications

Understanding latency is crucial, particularly in applications where timing is critical, such as remote surgeries. Latency, the delay encountered in the communication between a device and its target, is influenced by network data transfer rates and the volume of data being processed. In settings like remote surgeries, where low latency is essential, it allows surgeons to respond swiftly to live updates without compromising the quality of the video feed, crucial for making precise decisions during operations. This underscores the need for advancements in 5G technology to overcome such challenges, ensuring the seamless execution of highly sensitive tasks.

Historically, latency limitations have posed challenges. For instance, during pioneering remote surgeries in the early 2000s, such as those conducted by Dr. Marescaux and his team, the operational setup only allowed a maximum resolution of 1024x768 pixels at a 10 Mbps transfer rate, with the surgeries operating at an average latency of 155 ms, peaking at 330 ms in some cases [4]. This level of latency was barely sufficient and suggested that enhancements in video quality and data transfer rates were necessary to improve surgical outcomes.

Research by Zheng and colleagues highlights that typical response times for endoscopic surgeons in complex scenarios averaged around 397 ms [5]. However, with 5G, it is feasible to achieve much lower latencies, improving both the speed and quality of surgical video feeds, which is critical for remote operations.

Adoption of 5G networks across various countries has facilitated advancements in remote surgical technologies. This development has enabled the execution of remote robotic procedures using 5G networks [6]. For instance, in December 2018, a significant procedure involving a remote hepatectomy on a porcine model was carried out in Fujian, People's Republic of China. Utilising the Kangduo robotic surgery system and a 5G network provided by Huawei Technologies and China Unicom, the operation was conducted remotely with the control equipment located about 48 km away at the Fujian Branch of China Unicom. The surgeon successfully operated two robotic arms to perform the surgery, which lasted about an hour with minimal blood loss and an impressive latency of less than 150 ms [7].

In another groundbreaking event in 2020, a series of laparoscopic surgeries were performed in China using the “MicroHand” surgical robot, where the surgeons operated from Qingdao on porcine models located about 3000 km away in Anshun [8]. These procedures, which compared a traditional wired connection to a modern 5G network, demonstrated that 5G could achieve a lower latency of 264 ms, compared to 206 ms with the wired connection, enhancing the surgeon's ability to perform precise manoeuvres remotely. The surgeries concluded successfully, with total durations around two hours and minimal complications.

Moreover, the reduction in latency achieved through 5G networks not only enhances the mechanical aspects of remote surgeries but also facilitates the integration of AI technologies that depend on quick data processing.

1.1.3. AI and Medicine

In parallel with technological advances in 5G, the integration of Artificial Intelligence (AI) in medical applications, particularly surgeries, has made significant strides [9]. These AI systems, especially those enhanced by advancements in computer vision (CV) and machine learning (ML), have the potential to guide decisions during medical procedures. For this purpose, they will increasingly rely on the high-speed, real-time data transmission capabilities of 5G technology. This integration is crucial, as it allows AI to interpret and respond to complex surgical scenarios effectively, where every millisecond counts.

The advent of deep learning, especially through convolutional neural networks (CNNs), has revolutionised this aspect. CNNs, inspired by the human visual cortex, excel at recognising visual objects even under challenging conditions like variable lighting, obstructions like smoke and blood, or blurred imagery [10]. These networks can adapt and learn from visual disturbances common in surgeries, a significant leap over previous ML algorithms that failed under less-than-ideal conditions.

Further enhancements in AI's surgical applications came from integrating long short-term memory (LSTM) networks with CNNs [11]. While CNNs handle the visual recognition tasks, LSTMs add a crucial memory component, akin to a "memory cortex". This combination allows AI systems to understand sequences and contexts within surgical videos, rather than just static images. This is critical in surgery, where understanding the sequence of events can be as important as recognising the tools and actions at any given moment.

Current research is intensely focused on enabling AI systems to not only recognise different surgical phases and track the use of surgical tools but also understand their context and relevance within the ongoing procedure [12]. This capability is crucial, as it extends beyond mere identification to providing actionable insights that can guide surgeons in real-time, a process greatly enhanced by the low-latency transmissions offered by 5G networks.

Advancements in 5G and AI technologies have begun transforming medical practices, particularly through remote surgeries that combine robotics and AI. This integration has enabled not only remote operations but also the potential for AI-driven decision making in real-time medical contexts. However, to fully leverage AI in surgery, significant challenges remain. These include the need for vast and diverse datasets to train robust AI models, the development of sophisticated metrics for performance evaluation, and enhancing the explainability of AI decisions to ensure they are reliable and understandable in critical medical scenarios.

The synergy between 5G's rapid data handling and AI's advanced analytical capabilities is setting new standards in surgical precision and safety, showcasing the transformative impact of real-time technological integration in healthcare.

1.1.4. AI applications in Inspecting

While AI technologies have significantly advanced surgical practices, their transformative impact extends across various sectors, including manufacturing, infrastructure maintenance, and public safety. The integration of 5G plays a critical role here, enabling real-time data processing and decision-making crucial for AI-driven robotic inspections.

For predictive maintenance, industries increasingly leverage modern technologies equipped with sensors and advanced imaging tools. Image-based inspection is a key area where visual images captured from devices such as endoscopes and thermal imaging cameras play a pivotal role. These technologies allow inspectors to access and evaluate critical mechanical components that were previously challenging to examine. The low-latency communications provided by 5G networks are essential, enabling real-time transmission of high-volume visual data, which facilitates immediate analysis and actions.

Traditionally, image-based inspections relied on human judgement to identify faults. However, the advent of deep learning has shifted this toward automating these processes using AI. For example, CNNs have been employed to detect defects in photovoltaic cells with an accuracy of 93.02%, while R-CNNs have been used for automated, real-time object detection training [13]. This demonstrates AI's potential to transform predictive maintenance into a more precise and automated operation.

Research has also been directed toward reducing human risk and logistical burdens by aiming for more objective, automated, and repeatable inspection results. Innovations like mobile inspection robots and Unmanned Aerial Vehicles (UAVs) have been explored, though human oversight remains indispensable, often requiring on-site personnel for operational management [14].

To enhance objectivity and consistency in inspections, recent advancements in vision systems have been instrumental. Automated Visual Inspection (AVI) systems are increasingly utilised in infrastructure monitoring to ensure safety and minimise processing time [15]. These systems incorporate a multi-technology vision setup that operates under various environmental conditions, utilising algorithms for image enhancement and autonomous image processing using ML and DL techniques.

The principles of real-time data processing and rapid decision-making enabled by 5G in industrial inspections set the stage for broader applications in AI real-time decision-making from streamed data, crucial in various fields including robotics and emergency response.

1.2. Introduction to the Project

1.2.1. Research Questions

In this thesis, the primary focus is on leveraging 5G technology to enable real-time communication for deep learning applications in robotics, specifically for crack detection. While 5G technology has already proven capable of providing low enough latency for remote human-operated surgeries—demonstrating the feasibility of single-way communication—the complexity increases when considering bilateral communication and AI-driven decision-making. Therefore, the research aims to address the following questions:

1. Primary Research Question:
    - Can we build a real-time remote feedback-driven action system using 5G?
2. Secondary Research Questions:
    - Can we achieve bearable latency when the decisions are based on machine learning processing?
3. Further Discussion:
    - What is considered “real-time” for human needs, and how does this benchmark apply to the context of AI-driven robotic systems?

These questions guide this project in exploring the potential and limitations of 5G technology for "Realtime communication using 5G to enable deep learning on the cloud for robotic applications."

1.2.2. Main Objective: AI Real-Time Decision-Making from Streamed Data

Off-board processing could be a pivotal advancement to enhance the reliability and capabilities of machine learning models in real-time applications. This term refers to the capability of running an ML model on one device using data from another, particularly significant in robotics where data processing can be transferred from mobile devices to a standardised data analysis centre. This approach benefits from 5G's capability for rapid, reliable data transmission, allowing for real-time analytics and decision-making processes that are both efficient and scalable [16].

This approach allows robotic systems to operate with simpler hardware, reducing weight, volume, and cost, while extending battery life—crucial for applications requiring swift and dependable decision-making. Furthermore, the ability to execute deep learning models off-board significantly enhances and customises robotics control systems without direct hardware interaction. Robust, low latency 5G connections ensure seamless data exchange, essential for maintaining the integrity of real-time robot operations.

Furthermore, the off-board execution of deep learning models marks a significant advancement in robotics. It facilitates enhancements and customization of the control system through deep learning without requiring direct interaction with the robot. For this to be effective, a robust and reliable internet connection is crucial, allowing data exchange with minimal latency to ensure no vital information from the robot is missed. Given the capabilities of 5G networks to meet these requirements, one might wonder: Could we perhaps extend this to not just a unidirectional, but a bilateral data exchange that operates in real-time for a feedback-driven action system?

The potential for AI to analyse and act upon data significantly enhances efficiency and safety across various sectors. For example, in manufacturing, AI-driven robots equipped with advanced sensors and supported by 5G connectivity can perform precise and predictive maintenance [17]. This not only helps in preventing equipment failures but also reduces downtime and maintenance costs. Similarly, in civil engineering, AI aids in the early detection of structural weaknesses or damages, such as cracks or corrosion in bridges and buildings, ensuring timely maintenance and enhancing safety.

This development not only allows a device to seamlessly switch between various tasks but also bolsters its autonomous decision-making capabilities. The main requirement for these technologies is the development of a system capable of managing input, processing, and decision-making in a manner that meets real-time needs as perceived by humans, with sufficiently low latency to fulfil the specific requirements of each application.

The advancement of real time communication using 5G to enable deep learning on the cloud for robotic applications could redefine the potential of 5G. This dissertation explores these possibilities, particularly how they can enhance remote healthcare and other critical services, asking pivotal questions about maintaining low latency in vital real-time applications and the broader impact of AI integration on remote operations' effectiveness and reliability.

1.3. Structure of the Dissertation

This dissertation is organised into several key sections to comprehensively address these questions. Following this introduction, the literature review will provide a detailed examination of current technologies and their applications in 5G and AI, highlighting gaps and opportunities for innovation. The methodology section will describe the research approaches and data analysis techniques employed to investigate these technologies. Subsequent sections will present the results, discuss their implications in the context of existing technologies, and propose future directions for research. Finally, the conclusion will synthesise the findings and outline their significance for the advancement of 5G and AI in various applications.

2. Literature Review

2.1. Machine Learning Models for Crack Detection

In exploring machine learning applications for crack detection, a variety of methodologies have been identified, each tailored to specific types of data and detection needs. These methods range from convolutional neural networks, which are effective at feature extraction from complex image data, to support vector machines, noted for their precision in classification tasks. The synthesis provided by [18] highlights the evolution of these technologies, noting a shift towards integrated systems that combine multiple analytical techniques to enhance detection accuracy and efficiency. This comprehensive review underlines a trend towards more adaptive and robust crack detection solutions, setting a foundation for the proposed study's approach to leveraging deep learning for enhanced structural health monitoring.

Hafiz et al. provide a comprehensive review of diverse image processing-based methods for crack detection, illustrating the wide array of applications and their respective outcomes in various domains. They detail several innovative approaches, each tailored to specific types of infrastructure and materials, showcasing the adaptability and precision of modern crack detection technologies [18].

In the field of crack detection, convolutional neural networks (CNNs) are prominently utilized due to their robust feature extraction capabilities which are critical for the accurate identification and segmentation of cracks. These networks typically comprise three key layers: the convolutional layer that extracts pertinent features from images, the pooling layer that reduces image dimensions through down sampling, and a fully connected layer that classifies the image based on the extracted features [19].

Crack detection technologies have evolved to not only identify cracks but also classify and measure them, enhancing the diagnostic capabilities of these systems. For instance, GoogleNet CNN, a sophisticated architecture, has been adapted for crack classification, utilizing a feature pyramid network to enhance crack delineation. This method has proven effective, achieving a precision of 80.13%, although it requires approximately 16 seconds to process a high-resolution image (6000 x 4000 pixels), pointing to a trade-off between accuracy and processing time [20].

Furthermore, the deployment of CNN models extends beyond laboratory settings to real-world applications such as pavement and infrastructure inspection. A notable study utilized CNNs to detect defects in pavement images collected via smartphone sensors, achieving significant precision, recall, and F-measure scores of 0.8696, 0.9251, and 0.8965, respectively [21]. This underscores the practical applicability of CNNs in environments requiring rapid and reliable assessments.

For instance, K-means clustering combined with Gaussian models, as explored by Oliveira and Correia, offers an unsupervised learning approach that achieves high accuracy without the need for manual labelling of dataset images. This method is particularly effective for road image analysis, achieving notable F-measure and recall rates, but shows limitations in detecting very narrow cracks [22].

Landstrom and Thurley utilize logistic regression within a morphological image processing framework to detect major cracks in steel slabs, addressing the industrial need to prevent costly defects in production [23].

Furthermore, Yang et al.'s Feature Pyramid and Hierarchical Boosting Network (FPHBN) significantly advances pavement crack detection by integrating deep learning to enhance feature recognition and accuracy in complex image backgrounds [24].

Support Vector Machines (SVM) have been effectively used by Gavilan et al. to detect road distress with high precision, employing advanced image preprocessing techniques to refine the detection process [25].

Additionally, combinations of SVM with Random Forests have been applied to bridge inspections, as demonstrated by Prasanna et al., who developed a system that not only detects but also classifies cracks with high accuracy, contributing to the safety and maintenance of critical infrastructure [26].

The utilization of Artificial Neural Networks (ANN) and other neural architectures further highlights the shift towards more sophisticated, automated crack detection systems. Wu et al.'s ANN-based method, MorphLink-C, for example, addresses the fragmentation of cracks in images, significantly improving classification accuracy and training efficiency for road and pavement maintenance [27].

Each of these methodologies contributes uniquely to the field of crack detection, providing a broad spectrum of tools and techniques that enhance the reliability and efficacy of current practices. The continuous evolution of these technologies underscores the dynamic nature of machine learning applications in industrial and infrastructural maintenance.

2.1.1. Challenges of Real-Time Crack Detection

Currently, numerous researchers are investigating algorithms for detecting cracks, proposing various effective methods tailored to different scenarios. However, while these algorithms demonstrate high accuracy, many are impractical for deployment on mobile robots for real-time crack detection due to limitations in model parameters and substantial computational demands. This becomes particularly evident in scenarios requiring large-scale and swift crack detection, such as post-earthquake disaster assessments, where a real-time and relatively precise detection method is crucial. Achieving millimetre-level accuracy in crack width measurement is essential for structural health monitoring of concrete structures and rapid post-disaster damage assessments.

Nonetheless, attaining such precision from images captured by a mobile robot's camera, especially from a distance, poses challenges. Additionally, existing studies often rely on ideal image capture conditions, making it difficult to calibrate shooting distances for crack detection in practical settings. In case of drones, for example, the device might experience flight turbulence, and captured images are subject to environmental interferences, necessitating accurate crack width measurements under complex conditions. Furthermore, ensuring the safety of robots complicates efforts to maintain close proximity to buildings for extended periods, particularly during large-scale crack detection. Consequently, most methods require robots to maintain a certain distance from buildings, leading to decreased detection accuracy.

2.2. Diverse Tasks in Machine Learning

2.2.1. Binary Classification

Binary classification involves categorizing data instances into two groups, typically labelled as positive and negative [28]. In this context, a positive label might indicate the presence of a condition like an illness or abnormality, whereas a negative label signifies no deviation from the norm. Each classification outcome can be categorized into one of four types: true positive (TP), where the positive condition is correctly identified; true negative (TN), where the absence of the condition is correctly identified; false positive (FP), where a condition is incorrectly identified in a negative instance; and false negative (FN), where a condition is missed in a positive instance.

To evaluate the effectiveness of a binary classifier, a 2x2 confusion matrix is utilized, which tallies the occurrences of TP, TN, FP, and FN. This matrix is fundamental for calculating key performance metrics such as accuracy, sensitivity (also known as recall or true positive rate), specificity (true negative rate), and precision (positive predictive value). These metrics are defined as follows:

$$Acurracy = \frac{TP + TN}{TP + TN + FP + FN} \in [0, 1] \qquad Specificity = \frac{TN}{TN + FP} \in [0, 1]$$

$$Sensitivity\ (Recall) = \frac{TP}{TP + FN} \in [0, 1] \qquad Precision = \frac{TP}{TP + FP} \in [0, 1]$$

Equations 1 – Fundamental metrics for binary classification

While accuracy represents the proportion of correctly predicted instances among all evaluations, sensitivity and specificity provide more detailed insights into the model's performance, particularly when dealing with imbalanced datasets where the prevalence of positive versus negative instances varies significantly. Often, metrics like precision and recall or sensitivity and specificity are examined in tandem to gain a comprehensive understanding of a classifier's performance, emphasizing aspects beyond mere accuracy that are critical in fields such as medical diagnostics.

2.2.2. Image Segmentation

Image segmentation is a critical process in computer vision that involves partitioning an image into multiple segments or regions, allowing for easier identification and localization of different objects and their boundaries within the image [28]. This technique transforms an image matrix into a segmentation mask, where each pixel or voxel (in the case of 3D images) is classified into distinct categories. In binary image segmentation, the goal is to create a mask where the regions of interest are marked as '1' (positive) and all other areas as '0' (negative). More complex forms, such as semantic segmentation, use multiple integers to represent a variety of classes within the image.

Evaluating the effectiveness of segmentation masks typically involves various metrics, with accuracy being a common choice. For binary segmentation tasks, accuracy could theoretically be calculated by tallying the true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) across the image. However, this method often leads to skewed results, especially in applications like tumour segmentation in medical imaging, where the objects of interest (positive pixels) constitute a very small portion of the image compared to the background (negative pixels). Consequently, even a model that fails to accurately identify positive targets might still show high accuracy if it correctly identifies the vast majority of negative pixels.

To address this imbalance and provide a more meaningful assessment of model performance, evaluation often focuses less on the total number of correct negative predictions (TN) and more on the similarity between the predicted positive segments and the actual, human-annotated ground truths. This approach emphasizes the importance of accurately detecting and delineating the regions of interest, rather than merely recognizing the abundant negative space.

2.2.3. Object Detection

Object detection is a crucial task in image processing that involves identifying and localizing various objects within an image. This is typically achieved by drawing bounding boxes around each object and classifying them into different categories [28]. A proficient object detector not only identifies all relevant objects in the image but also minimizes false detections, accurately places bounding boxes around the objects, and correctly classifies each object.

Evaluating object detectors involves several layers due to the complexity of the task. Initially, the assessment begins by identifying the number of objects detected within a specific class. A critical factor in this evaluation is determining how closely a predicted bounding box needs to align with a ground-truth box for it to be considered a successful detection. The Intersection over Union (IoU) is commonly used to measure this; a prediction is deemed accurate if the IoU between the predicted and ground-truth boxes surpasses a certain threshold, typically set at 0.5. If multiple predicted boxes meet this IoU criterion with the same ground-truth box, only the one with the highest IoU is considered a true positive (TP), while the others are classified as false positives (FP).

To quantify the effectiveness of an object detection model, precision and recall metrics are calculated based on the counts of TP, FP, and false negatives (FN) — where FN represents the number of ground-truth boxes that did not have a corresponding accurate prediction. Object detectors also generate a confidence score for each bounding box, reflecting the model's certainty about the prediction. By adjusting the confidence threshold, the balance between precision and recall can be shifted, influencing the overall performance metrics.

The precision-recall curve (PRC) is then plotted by varying the confidence thresholds, and from this curve, the Average Precision (AP) for each class is derived as the area under the curve. The overall performance of the model is evaluated using the mean Average Precision (mAP), which averages the APs across all classes. Different IoU thresholds can be applied to assess the model's accuracy more rigorously, such as mAP@0.75 or mAP@0.9, where a higher threshold like 0.9 demands greater precision in bounding box placement, making it suitable for applications requiring exact object localization [29].

2.3. Real-Time Data Streaming Technologies

2.3.1. Importance of live streaming for Robotics

Streaming technologies are pivotal in robotics for enabling real-time data communication, essential for tasks such as navigation, object recognition, and interactive operations. The ability to stream information continuously helps robots to perform tasks efficiently in dynamic environments. For instance, in surgical robots or those involved in disaster response, the immediacy of streaming can be the difference between success and failure, making the reliability and efficiency of these protocols fundamentally important [30].

2.3.2. Challenges and Parameters of Streaming Protocol

The integration of streaming protocols in robotics, particularly in cloud robotics, introduces complex challenges and requires careful consideration of several parameters. Cloud robotics benefit significantly from computation offloading, which involves transferring intensive computation tasks from the robot to the cloud. This process, however, is not without its difficulties. The primary challenge lies in managing the high latency and potential unreliability of network connections that are essential for real-time data transmission. Essential parameters that need meticulous planning include the volume of data transferred, the computational capabilities of the robot, network delays, energy consumption, and task completion deadlines. These factors are critical as they directly impact the Quality of Service (QoS) and the overall functionality of robotic systems [31].

2.3.3. Overview of Live Stream Protocols and Their Applications

Streaming protocols vary widely in their design and application, each catering to specific needs in the streaming ecosystem:

- RTMP (Real-Time Messaging Protocol), developed by Macromedia and later acquired by Adobe, has historically been crucial for streaming audio, video, and data over the Internet. Despite its decline due to the phasing out of Flash, RTMP is still essential for ingesting live streams into platforms via RTMP-enabled encoders. It is known for its low latency but is not optimized for scalability or viewer experience.
- RTSP (Real-Time Streaming Protocol) is predominantly used in surveillance and CCTV systems due to its effectiveness in controlling media streaming sessions. It supports low-latency operations and is commonly used in IP cameras, making it ideal for applications requiring precise control over streaming.
- WebRTC (Web Real-Time Communication) offers ultra-low latency communications directly between browsers and devices, which is crucial for real-time robotic controls and interactions. Its ability to function without plugins and across different platforms and devices enhances its utility in modern robotic systems.

- HLS (HTTP Live Streaming) and SRT (Secure Reliable Transport) are other notable protocols, with HLS being widely adopted for its reliability and SRT emerging as a solution for high-quality, low latency streaming over unpredictable networks.

For the purposes of this thesis, WebRTC was selected as the preferred streaming protocol due to its strong performance in scenarios requiring ultra-low latency. In cloud robotics, where immediate reaction times are crucial for tasks like real-time mapping and autonomous navigation, WebRTC's capability to provide near-instantaneous communication is invaluable. Unlike RTSP, which may introduce slight delays due to its session-based nature, WebRTC facilitates a more direct and faster data transfer, which is crucial for maintaining the high standards of QoS necessary in robotic operations. This choice is further supported by WebRTC's robust handling of video and audio streams, which are integral to the operation of advanced robotic systems [32].

2.4. WebRTC and Its Mechanisms

Web Real-Time Communication (WebRTC) is an innovative web technology that enables direct audio and video communications, chat, and peer-to-peer file sharing directly within web browsers and mobile applications, without requiring external plugins. Launched as an open-source initiative by Google in 2011, WebRTC is supported by major browsers such as Google Chrome, Mozilla Firefox, and Opera [33]. The technology consists of several key components accessible via a JavaScript API, which are crucial for developers creating real-time interactive web applications. These components include:

- MediaStream: Accesses device cameras and microphones for live media capture.
- RTCPeerConnection: Establishes audio and video connections between browsers.
- RTCDataChannel: Facilitates direct data transfer between browsers.

WebRTC is a collection of APIs and protocols defined by entities such as the W3C and IETF, which enhances its support across different platforms and browsers.

2.4.1. Signalling in WebRTC

Signalling, the process of coordinating and controlling communication sessions, is central to WebRTC but is not specified within the WebRTC standard, allowing flexibility in integration with existing systems. Signalling protocols such as SIP, XMPP, and WebSocket can be employed, with WebSocket being a popular choice due to its ability to maintain open, bi-directional communication streams, which are essential for real-time interactions. In practice, signalling involves several stages:

- Exchange of Control Messages: These messages manage session setup and include commands like “initialize” and “initiator”.

- Use of Session Description Protocol (SDP): SDP messages exchange media capabilities and configurations between peers.
- Establishment of Peer-to-Peer Connection: This follows the successful negotiation of session parameters.

These steps ensure that the necessary configurations are in place before streaming begins. The use of JSON for signalling data allows for lightweight and efficient transmission. This structured signalling process is vital for leveraging WebRTC's capabilities to deliver high-quality, real-time communication experiences.

2.4.2. Server Application

For WebRTC implementation, the server application can operate as a WebSocket server. Traditionally, Node.js is used due to its event-driven architecture and non-blocking I/O capabilities, which enhance the application's scalability and throughput. However, other platforms and languages can also be used for this purpose.

The WebSocket server manages communications and typically handles various types of messages critical for establishing and maintaining communication between clients. These include control messages, media information messages (SDP offer and SDP answer), and network information messages (ICE candidates). The server also tracks client connections, determines session initiators, and removes clients as needed when sessions conclude.

2.4.3. Client Application

On the client side, the application leverages the WebRTC API to set up a peer-to-peer connection essential for direct media exchange. The process begins with the exchange of local and remote descriptions containing audio and video media information via the signalling server.

The sequence of the signalling process is as follows:

- Offer Creation: One peer initiates the connection by creating an offer containing its Session Description Protocol (SDP) information.
- Offer Transmission: This local description is sent to the other peer via the signalling server.
- Answer Process: Upon receiving the offer, the other peer acknowledges the offer as its remote description, generates an answer, sets its own local description, and sends this back to the initiating peer.
- Session Establishment: When the initiating peer receives the answer, it sets it as the remote description.

This structured interaction ensures that both clients are synchronized with each other's media and network configurations, facilitating a robust peer-to-peer communication channel. This setup process is critical for the real-time, efficient transmission of audio and video data required in WebRTC applications.

2.4.4. AIORTC: WebRTC in Python

While Node.js is a popular choice for implementing WebRTC signalling servers, Python provides a powerful alternative through libraries such as AIORTC [34]. Implementing this solution follows similar principles to the traditional JavaScript approach. Initially, the AIORTC library must be installed through a package manager to utilize it within the project. Subsequently, a signalling mechanism is established, which is crucial for exchanging Session Description Protocol (SDP) messages and Interactive Connectivity Establishment (ICE) candidates between peers. This can be accomplished using protocols like WebSockets or HTTP, facilitating the necessary communication for establishing connections.

While the procedural steps for connecting peers and managing media streams and data channels using AIORTC are similar to those in the JavaScript implementation, AIORTC offers several distinct advantages. Firstly, AIORTC leverages the extensive Python ecosystem, which includes a wide range of libraries and tools for data analysis, machine learning, and scientific computing. This makes it particularly suitable for applications that require integration with other Python-based systems. Moreover, Python's syntax is often considered more readable and easier to learn compared to JavaScript, simplifying the development process, especially for developers who are more familiar with Python [35].

Additionally, AIORTC can be integrated with various Python frameworks and libraries, providing greater flexibility in building complex applications. This includes integration with asynchronous programming frameworks like asyncio [36], which can enhance the performance of real-time applications. AIORTC also ensures compatibility with various operating systems, including Windows, macOS, and Linux, making it essential for developing applications that need to run on multiple environments.

Furthermore, Python's performance in handling asynchronous tasks can be advantageous in real-time communication scenarios. AIORTC is designed to efficiently manage the asynchronous nature of WebRTC signalling and media handling. Finally, the extensive documentation, tutorials, and community support available for Python can be particularly beneficial when troubleshooting issues or seeking optimization techniques. This robust community support ensures that developers have access to a wealth of resources, aiding in the successful implementation of WebRTC solutions using AIORTC.

2.5. Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a versatile statistical method used to reduce the dimensionality of a data table, retaining its essential features. This method transforms the original variables into new uncorrelated variables called principal components, which account for most of the variance in the data. By focusing on these principal components, PCA simplifies the data structure, making it easier to analyse and visualize complex datasets [37].

2.5.1. Theoretical Foundation of PCA

PCA relies on several key mathematical concepts to achieve its objectives. Dimensionality refers to the number of characteristics or parameters in the data. High-dimensional data, often encountered in large datasets, can be challenging to interpret. PCA addresses this by reducing the number of variables while preserving the essential information through linear combinations of the original variables.

Another fundamental concept is correlation, a statistical measure that indicates the strength and direction of the linear relationship between variables. PCA uses the covariance matrix, which captures these correlations, to transform the data into principal components. These components are orthogonal to each other, meaning they are uncorrelated and contain distinct information about the dataset. This orthogonality ensures that the principal components provide a comprehensive yet non-redundant summary of the data.

Eigenvectors and eigenvalues are central to PCA's mathematical process. Eigenvectors represent directions in the data space, while eigenvalues indicate the magnitude of variance along those directions. By decomposing the covariance matrix into its eigenvectors and eigenvalues, PCA identifies the principal components that explain the most variance in the data.

2.5.2. Importance and Applications of PCA

The primary advantage of PCA is its ability to simplify complex datasets by reducing dimensionality, which helps in uncovering underlying patterns and relationships. This simplification is particularly useful in fields like image processing, genomics, and finance, where large datasets are common. PCA's graphical results, often displayed as biplots, provide intuitive visualizations that aid in the interpretation of data.

Moreover, PCA is widely accessible and easy to implement using programming languages such as Python, thanks to libraries like NumPy and scikit-learn. These tools provide efficient functions for performing PCA, making it a popular choice for data scientists and researchers. The accessibility of PCA through such tools has contributed to its widespread adoption across various disciplines, further highlighting its versatility and utility.

2.5.3. PCA in Crack Detection for Robotic Application

In the realm of robotic applications, particularly for crack detection, PCA can significantly enhance the effectiveness of machine learning (ML) segmentation masks. When an ML model segments an image to identify cracks, PCA can be applied to the segmented regions to determine the principal direction of these cracks. This is achieved by analysing the segmented data to find the directions that capture the most variance, corresponding to the primary orientation of the cracks.

By integrating PCA with ML segmentation, the robotic system can utilize this directional information to navigate and inspect the affected areas more effectively. For instance, in infrastructure maintenance, understanding the direction in which a crack propagates is crucial for assessing the severity of damage and planning appropriate interventions. PCA provides a robust method for extracting this directional information, thereby improving the accuracy and efficiency of robotic inspections.

In practice, the data table comprising observations and variables is pre-processed, centred, and possibly standardized before applying PCA. The singular value decomposition (SVD) of the data matrix yields the principal components, which are then used to determine the primary directions of the cracks. This process not only enhances the crack detection capability but also supports real-time decision-making, essential for autonomous robotic systems.

2.6. Comparative Analysis of 5G and Wi-Fi for Real-Time System

One of the main components required to establish a real-time system is the network. As many solutions today operate under Wi-Fi, it is essential to understand the performance differences between Wi-Fi and 5G technologies. Traditional Wi-Fi, which operates under standards such as IEEE 802.11n (Wi-Fi 4) and IEEE 802.11ac (Wi-Fi 5), provides adequate performance for many home and office environments. However, these earlier versions of Wi-Fi fall short when compared to the capabilities of Wi-Fi 6 and 5G, which cater to the growing demands for higher data rates and more reliable connections.

2.6.1. Wi-Fi Technology Evolution

Wi-Fi 6, also known as IEEE 802.11ax, represents a substantial advancement in wireless technology. It operates in the 2.4 GHz and 5 GHz frequency bands, with the extended version, Wi-Fi 6E, adding the 6 GHz band. Wi-Fi 6 can achieve impressive speeds of up to 9.6 gigabits per second (Gbps) in controlled settings, making it a robust choice for high-demand applications. These improvements are driven by features such as Orthogonal Frequency-Division Multiple Access (OFDMA), Target Wake Time (TWT), and improved beamforming, which enhance efficiency, battery life, and performance in dense environments [38].

2.6.2. The 5G Technology

In contrast, 5G, the fifth generation of mobile network technology, offers significant advantages over traditional Wi-Fi. 5G technology can achieve data transfer speeds of up to 20 Gbps, making it a superior option for applications that require ultra-low latency and high-speed data transfer. It operates in multiple frequency bands, including sub-1 GHz, 1-6 GHz, and millimetre-wave (24-100 GHz) bands, providing extensive coverage and high capacity. 5G's network slicing capabilities allow it to create virtual networks tailored to specific needs, enhancing its versatility for different applications.

2.6.3. Performance Comparison

The primary advantage of 5G over Wi-Fi 6 lies in its speed, latency, and reliability. While Wi-Fi 6 can reach speeds of up to 9.6 Gbps, 5G can achieve up to 20 Gbps. Moreover, 5G offers lower latency, which is crucial for real-time applications such as autonomous vehicles, remote surgery, and real-time gaming. The reduced latency of 5G, which can be as low as 1 millisecond, allows for near-instantaneous communication between devices, reducing the time delay experienced in data transmission [39].

For real-time systems, such as those used in this project, the choice between 5G and Wi-Fi can significantly impact performance. While Wi-Fi 6 provides substantial improvements over its predecessors and is suitable for many high-demand applications, 5G offers the additional benefits of higher speeds, lower latency, and more reliable connections. These features make 5G particularly advantageous for applications that require rapid response times and consistent data transfer rates.

2.7. Synchronising Devices with Precision Time Protocol (PTP)

The Precision Time Protocol (PTP) offers a cost-effective solution for aligning clocks on devices connected by a packet-based network. Instead of equipping each device with an expensive and highly accurate time source like an atomic clock or a GPS-equipped receiver, PTP allows for synchronisation using more affordable crystal oscillator-based clocks. PTP was designed to meet the stringent synchronisation requirements of various applications, including power systems, industrial automation, telecommunications, and manufacturing [40].

2.7.1. How PTP works

PTP operates at the application layer, where one or more devices act as the master clock, serving as the time reference. This master clock should have a reliable time source. The other devices, referred to as slave nodes, align their clocks to the master clock. Each PTP node has one or more ports and communicates with other PTP nodes in the network by implementing the PTP synchronisation protocol. The main steps of the PTP synchronisation protocol are as follows:

1. The master node periodically transmits a Sync message to the slave nodes, containing the sent time T1.
2. Optionally, a Follow_up message, which also contains T1, is sent depending on the timestamp processing mechanism.
3. When the Sync message arrives at a slave node, the received time T2 is recorded.
4. The slave node then sends a Delay_req message to the master clock, recording the time T3 locally.
5. The master node records the reception time T4 and sends it back to the slave node in a Delay_resp message.

The offset time Θ_k at a given slave node k is approximated by the formula:

$$\Theta_k = \frac{D1 - D2}{2}$$

where (D1 = T2 - T1) and (D2 = T4 - T3) correspond to the downstream (master to slave) and upstream (slave to master) delays, respectively. The slave node adjusts its local time by subtracting Θ_k to correct for the clock offset. This approximation works well if the delays D1 and D2 are symmetrical; however, asymmetrical delays can degrade synchronisation accuracy.

3. Methodology

3.1. Project Setup and Hardware Integration

Before selecting an appropriate crack detection model, the first step was to decide on the project setup elements. In this sense, embedded computing boards are essential for prototyping, and the Nvidia Jetson series stands out due to its high performance and compact size. These boards have gained popularity not only for their robust capabilities but also for their seamless integration with flight and motor controllers such as the Pixhawk, making them ideal for drones and rovers. For this project, the Jetson NX Orin 16GB was chosen. This board provides impressive specifications, including 16GB of 128-bit LPDDR5 memory with a bandwidth of 102.4GB/s, a GPU with a maximum frequency of 918 MHz, and a CPU with a maximum frequency of 2 GHz. Measuring 110mm x 110mm x 71.65mm, it is compact yet powerful [41].

In contrast, to represent a less powerful board that relies on off-board processing, an HP laptop was included in the setup. This laptop features an AMD A4-5000 APU with four CPU cores running at 1.5 GHz. The 28-nanometer chip integrates a Radeon HD 8330 GPU with 128 shaders, based on the GCN architecture and clocked at 500 MHz. It also includes a single-channel DDR3(L)-1600 memory with 4.0 GiB of RAM. The performance difference is significant: the Jetson NX Orin's GPU, with its higher clock speed and more advanced architecture, offers substantially better performance, making it approximately 8 times faster in processing intensive tasks compared to the Radeon HD 8330 GPU. Similarly, the Jetson's CPU, with a maximum frequency of 2 GHz, operates at a higher speed than the 1.5 GHz APU of the HP laptop, providing roughly 33% faster processing power.

For streaming, the Oak-1-max camera was selected due to its compact dimensions (36 mm x 54.5 mm x 27.8 mm), lightweight nature (53.1 grams), and excellent resolution of 32MP (5312x6000). The camera's extensive settings and easy integration with Linux OS via its SDK made it highly suitable for real-time applications.

The central processing unit, or host device, in this setup was a Lenovo Legion 5. It is equipped with an Intel Core i5-10300H CPU running at 2.50 GHz and 16.0 GB of installed RAM. The system operates on a 64-bit operating system with an x64-based processor. Additionally, it features an Nvidia RTX 2060 GPU with DirectX version 12.0, 1920 CUDA cores, a core clock of 1200 MHz, and a memory data rate of 10.96 Gbps. The GPU’s memory interface is 192-bit, providing a memory bandwidth of 263.04 GB/s. It has 14289 MB of total available graphics memory, including 6144 MB of dedicated video memory and 8145 MB of shared system memory. This robust configuration ensures efficient handling of intensive processing tasks.

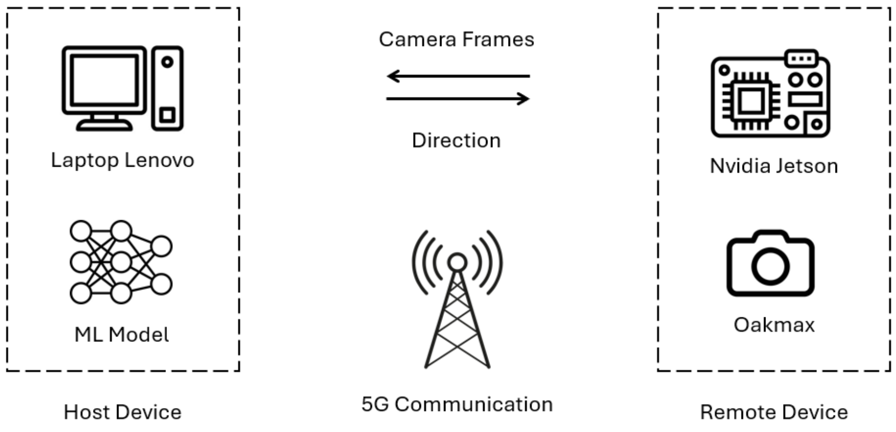


Figure 1: Enhanced System Setup

The integration of all hardware components and software was streamlined through Python, owing to its robust libraries and an active community that provides extensive support through forums and documentation. Python's ecosystem simplifies the development process on Linux OS, which is supported by Jetson boards. Linux offers a stable and versatile development environment, aligning perfectly with the need for a reliable system capable of continuous upgrades and modifications.

3.2. Selection and Validation of a ML Model for Crack Detection

To validate the feedback-driven actions based on deep learning, a crack segmentation model was selected. Given the nature of crack formation and distribution, the host device can suggest the direction for the remote device to follow to maintain the crack's trajectory. Among the numerous models available, the chosen model stood out due to its performance in detection and segmentation as well as its straightforward integration with the local device.

The model underwent training on a highly augmented subset of the "Roboflow" computer vision dataset, which ensured robustness and generalization. The training set comprised 91% of the data, supplemented by various augmentations such as flips, rotations, shears, exposure adjustments, and noise addition. These augmentations were designed to mimic real-world variations in input data, thus enhancing the model's adaptability and reliability.

The evaluations conducted over Google Colab demonstrated the model's strong performance metrics. The table below summarises the results:

Metric	Object Detection	Image Segmentation
Precision (P)	0.627	0.544
Recall (R)	0.576	0.576
Mean Average Precision (mAP50)	0.628	-
Mean Average Precision (mAP50-95)	0.398	-

Computational Efficiency	141.9 GFLOPs	141.9 GFLOPs

Table 3.2-1: Performance Metrics of the Machine Learning Model.

These metrics indicate the model's high degree of accuracy in identifying and segmenting relevant objects within images. The mean Average Precision (mAP50) and mAP50-95 reflect the model's comprehensive detection capabilities across varied object sizes and shapes. Additionally, the model's computational efficiency, requiring 141.9 GFLOPs, aligns well with the processing capabilities of our local devices, making it suitable for real-time applications.

The decision to use this particular model was further justified by its seamless integration into our existing systems. This integration was facilitated by the ability to utilize pre-existing trained weights, a feature critical for fast-tracking deployment and reducing additional computational overhead. Moreover, the model's architecture supports on-device inference, which is crucial for our application's need for real-time processing capabilities without relying on continuous cloud connectivity. The high precision and recall rates ensure that the model not only detects but accurately segments objects in real time, a critical requirement for the success of our application.

3.3. Establish Callback Action Based on ML Output

The primary objective of the system was to determine the next direction for the robot's trajectory based on the output from the crack segmentation model. This capability is crucial for the development of autonomous devices, as it enables a robot to navigate and track cracks by making real-time decisions based on current inputs. Minimizing latency is essential due to the risks associated with inertial movements, where the robot continues to move based on a previous command rather than responding to updated environmental inputs.

With this aim, the machine learning model output was investigated to understand how the detection and segmentation outputs were generated by the model. This phase presented challenges as the outputs required a series of data manipulation and conversions to allow post-processing. The key challenges regarding data manipulation and the development of a callback-action algorithm are discussed in the following sections.

3.3.1. Conversion and Normalization

The transformation of data began with converting standard image data arrays into tensors, which are more suitable for processing on graphics processing units (GPUs). This conversion was essential for harnessing the computational power of GPUs, which significantly accelerates machine learning tasks. Additionally, the image data was normalized, adjusting the range of pixel intensity values to a scale that neural networks process more effectively. This step is fundamental as it enhances the model’s ability to learn from the data more efficiently and produce more accurate predictions.

3.3.2. Reformatting for Analysis

After the initial processing in the GPU environment, the data had to be converted back into a simpler array format to facilitate further analysis. This step involved transferring the data from the GPU back to the central processing unit (CPU), a necessary move to leverage certain analytical libraries that operate in the CPU environment. This series of data transformations and analyses ensures that the system can quickly and accurately interpret crack data to adjust the robot's path in real-time.

By leveraging GPU capabilities throughout the process, the system maintains high computational efficiency, which is crucial for real-time applications. This ensures that the robot can make timely decisions based on the most current data, thereby enhancing its ability to navigate and track cracks autonomously.

3.4. Linearization and Principal Component Analysis (PCA) Method

In developing an effective crack detection system, various mathematical models were initially considered for analysing the spatial distribution of detected cracks. Linearization models, such as the Mean Square Method (MSM), were among the first to be explored. These methods attempt to fit a linear regression model to the data points representing cracks, theoretically simplifying the direction determination process. However, these linear models proved unsuitable due to the inherent non-linear nature of crack paths, which often exhibit complex, multi-directional spreads that cannot be accurately captured with a simple linear approach.

As a result, Principal Component Analysis (PCA) was employed as a more robust alternative. PCA is a statistical technique used in exploratory data analysis and predictive modelling, involving the computation of eigenvalues and eigenvectors of a dataset to reduce dimensionality while preserving as much variability as possible. In the context of crack detection, PCA was utilized to determine the principal direction of crack propagation. This method accurately captures the predominant direction of the detected cracks and minimizes the impact of outlier data points that do not align with the main crack direction.

The computational advantages of PCA include its ability to efficiently reduce the number of dimensions without significant loss of information, which is particularly beneficial for real-time processing. The eigenvectors corresponding to the largest eigenvalues represent the directions along which the data is most spread out, providing a reliable indication of the crack's main trajectory. This makes PCA an invaluable tool in the accurate and efficient analysis of crack patterns, directly informing the robotic navigation system of the most probable path of crack progression.

3.5. Max Pooling Method

To enhance the computational efficiency of our crack detection system, a max pooling operation was integrated into the data processing pipeline. Max pooling is a sample-based discretization process commonly used in machine learning to down sample an input representation, reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned.

This technique was applied to the segmentation matrix, where it helped reduce the resolution of the crack detection output while preserving the most salient features—namely, the detected cracks. By applying the max pooling operation, only the maximum value within a predefined window (pooling size) is retained. This approach effectively reduces the computational load by decreasing the amount of data to be processed while maintaining the structural integrity of the crack paths within the matrix.

The selection of the pooling size was a critical decision that balanced computational efficiency and resolution of the detected features. A larger pooling size reduces more data but can potentially lead to the loss of critical details about the crack's path. Conversely, a smaller pooling size retains more detail but less significantly reduces computational demands. After several trials and adjustments, a factor of 16 was chosen, as it adequately preserved the critical details necessary for accurate crack path prediction while substantially reducing the computational burden.

The downscaled matrix retained enough detail to allow for the effective application of PCA, ensuring that the directionality determined by PCA remained unaffected by the reduction in resolution. This synergy between max pooling and PCA ensured that the system maintained high processing speeds without compromising the accuracy needed for real-time robotic navigation based on crack detection.

3.6. Crack Segmentation Tracking by Feedback Instructions

Once the PCA method provided the trend of the crack dispersion, the next step was to develop a straightforward algorithm to instruct the remote device on tracking the crack. As the focus of this thesis is not on advanced tracking algorithms, the proposed solution involves aligning the device with the centre of the trend line. Essentially, this process calculates the distance between the centre of the frame (CF) and the centre of the trend line (CTL). If this distance exceeds 10% of the frame length, the device is directed to move toward the CTL. Conversely, if the device is already aligned with the CTL, the instruction is to follow the southernmost and easternmost points of the trend line.

Combining PCA and the tracking algorithm enabled the establishment full-duplex communication locally through WebRTC. The next step involved expanding data exchange between different devices, transitioning the system to operate over Wi-Fi.

3.7. Assembly and Performance of the Remote Device

Considering the assembly of robots, the performance of NVidia’s development boards is notable. These powerful and compact boards can deliver excellent performance in machine learning tasks, working alongside the flight or driver controller, and easily integrating with cameras and sensors. This capability makes it feasible to implement segmentation models directly on these boards to create an autonomous device. However, under this approach, the robot’s performance relies on the board's capability to handle the ML chosen model. Furthermore, this can pose challenges for a generic robot solution intended for diverse applications.

To address these challenges, this project decentralized the performance limitation from the device’s board and distributed it to both data transmission and the central processing unit, referred to here as the remote device. To validate this approach, the remote device was composed of two different setups, switching between the Nvidia Jetson Orin and an HP laptop, both of which were described previously. This strategy was designed as a general solution for multiple applications, capable of operating with real-time response depending on the End-to-End (E2E) latency. This specification is intimately associated with the quality of the robot network.

As embedded computing boards have well-developed integration with Wi-Fi, the use of remote communication with other devices is popular among drones and rovers. Whether sending or receiving data, this technology faces strict limitations on the range of the Wi-Fi signal (around 30 meters) and the quality of the connection in terms of speed, stability, and interference. Therefore, for long-distance applications, 5G technology can be a reasonable solution. However, for shorter ranges, it is essential to have a solid comparison between both technologies to determine the most effective solution. Thus, this project aimed to compare latency under both Wi-Fi and 5G networks.

Playing an important role in reducing latency, the centre of processing is conceived as a computer capable of executing the machine learning model using inputs received from the remote device. This central unit, being static, can be equipped with dedicated eGPUs, high-speed CPUs, extensive RAM availability, and other advanced hardware components. Due to the lack of limitations regarding power supply, volume, and weight, this system can execute tasks with better performance than common embedded computing boards. Moreover, by offloading the heavy computational demands of ML models, the boards can better perform operational tasks such as processing sensor data and acquiring camera images. For this project, the central processing unit was a Lenovo laptop, which, although not powerful, provided reasonable performance for the required tasks.

3.8. Establishing WebRTC Connection Between Devices

To implement off-board real-time computing, whether under Wi-Fi or 5G, the key point is to ensure a high-quality connection between the devices. Reducing computational latency alone is insufficient if the transmission process involves long processing times and complex protocols. Therefore, it was imperative to research and implement an adequate streaming method, evaluating established protocols that could minimize latency while maintaining high resolution.

After careful analysis, WebRTC was selected for its low-latency streaming capabilities that rely on peer-to-peer technology. The open-source nature of WebRTC allows for extensive customization to meet specific project requirements, particularly its capability to support real-time latency, ensuring that the video broadcast reaches viewers without perceptible delays and with excellent video quality.

Consequently, a stable streaming configuration using the WebRTC protocol was established over Wi-Fi using Python and the AIORTC library. The procedure was divided into three distinct scripts: one for the signalling server, another for the remote device, and the last for the host device (the centre of processing). The server handles the signalling step, gathering ICE candidates from both remote and host devices, checking the connection, and determining the type of streaming and its parameters. This server facilitates the discovery and negotiation process necessary for establishing a peer-to-peer connection between the devices, ensuring that both devices are aware of each other’s network capabilities and streaming preferences.

Several libraries were necessary to support the connection, with AIORTC being the most crucial for establishing real-time communication. When using external libraries, it is essential to examine their documentation to understand how the procedures occur internally. In the case of AIORTC the most relevant aspect was the locking of both frames per second (FPS) and image resolution during transmission. This discovery was fundamental for modifying the transmission parameters and analysing its performance under different settings. Since the aim of this project was to reduce latency while maintaining quality in bilateral communication, the choice of image resolution depended on the machine learning model's ability to provide reliable segmentation for the chosen frame dimensions.

3.9. Off-Board ML Processing Over Wi-Fi

As the project also aims to explore feedback-driven action performance over 5G, establishing a reliable baseline over Wi-Fi is essential for comparative analysis and understanding the enhancements offered by 5G technology. Thus, the next step involved implementing bi-directional communication using the WebRTC protocol running under Wi-Fi. The setup was composed of the host device and the remote device, appearing in both configurations (Jetson Orin and HP laptop). Since the WebRTC protocol had been established locally, it was only necessary to connect both devices to the same Local Area Network (LAN) and change the WebSocket (WS) responsible for hosting the signalling server to the host device's IP address.

Upon successfully establishing the connection, the complete bi-directional communication was implemented as follows:

1. Remote Device: The script was designed to acquire frames from the camera using the Oak SDK and transmit them to the host device.
2. Host Device: The host device inputs the received frames into the pre-loaded machine learning model.
3. Processing: The segmentation mask output by the model is downscaled using max pooling and then processed using Principal Component Analysis (PCA) to determine the best-fitting line for the non-zero elements. This line represents the crack direction trend, providing a clear path for the remote device to follow.
4. Directional Instruction: For simplification, the host device sends a directional instruction as a normalized vector.

This integrated approach ensures that the remote device and the central processing unit work in tandem to deliver real-time performance, leveraging the strengths of both local and remote computational resources. By balancing the load and ensuring efficient data transmission, the system can achieve the necessary responsiveness for autonomous robotic applications.

3.10. Upgrade to 5G for Enhanced Data Transmission

One of the significant advantages of 5G over Wi-Fi is its less restricted range of transmission between remote and host devices. In urban areas and parts of the countryside where 5G technology is widely implemented, mobile robots could explore new possibilities for off-board autonomous applications. For instance, drones could analyse large structures that are challenging for human inspection. Similarly, autonomous systems could be deployed in rural areas, such as farms, to monitor crops or in forests to track deforestation activities.

Moreover, 5G is associated with more reliable connections. This reliability is crucial even for local applications, such as warehouse stock analysis, where consistent and uninterrupted connections are necessary to avoid data loss and ensure seamless operations. The enhanced reliability of 5G can prevent connection breaks, providing a stable communication link that supports various industrial and commercial uses.

These advantages make 5G a practical solution for advanced applications that require high-speed, low-latency, and reliable communication. This technology enables innovative uses, such as the off-board feedback-driven system proposed by this project, where rapid response to real-time changes is essential for accurate data processing.

3.11. Selection of the 5G Module

For this project, the 5G module “RM502-QAE” from Quectel was chosen due to several key features that made it highly suitable for integration. The RM502-QAE is optimized specifically for IoT and enhanced Mobile Broadband (eMBB) applications, making it an ideal choice for our needs. It adopts the 3GPP Rel-15 LTE technology and supports both 5G NSA (Non-Standalone) and SA (Standalone) modes, ensuring compatibility with a variety of network infrastructures.

One of the main reasons for selecting the RM502-QAE is its comprehensive documentation, which facilitates smooth integration and troubleshooting. This documentation is crucial for developers, providing detailed guidelines and support throughout the implementation process. The module includes a rich set of Internet protocols and industry-standard interfaces, making it versatile for a wide range of applications. Additionally, it supports USB and PCIe drivers for multiple platforms, including Windows, Linux, and Android, extending its compatibility across various systems.

The RM502-QAE's robust features and extensive support make it an excellent choice for ensuring reliable and efficient 5G connectivity. Its ability to operate seamlessly across different platforms and network modes provides the flexibility needed for the diverse requirements of this project.

3.12. Application of PTP in Off-Board Machine Learning Processing

In the context of this project, PTP was employed to synchronise the devices involved in off-board machine learning processing. This was essential to accurately measure processing times and ensure timely responses. The use of PTP allowed for precise alignment between the remote device (such as a robot or sensor) and the central processing unit, which handles the machine learning tasks.

To achieve the highest precision in synchronisation, the devices were connected using an RJ45 cable with a cross-over connection. This physical setup minimised network-induced delays and provided a stable connection. While hardware-based precision timing was not feasible due to cost and complexity, software-based precision through PTP provided sufficient accuracy for the project’s requirements.

The integration of PTP in this setup involved configuring the network settings to enable PTP on both devices. The master clock was established on the central processing unit (host device), while the remote device operated as the slave node. Through this setup, the system could maintain synchronised time, thereby ensuring that the machine learning model's outputs were processed and acted upon without significant delay.

3.13. Glass-to-Glass Measurement Implementation

Several methods to measure Glass-to-Glass (G2G) delay in video transmission have been proposed. For this project, we adopted a method similar to those described by Hill et al.[42]. This method involves presenting a running clock on a computer screen, filming it, transmitting the video, and displaying it through the video transmission system. Both the real clock and the clock displayed by the video transmission system are captured simultaneously by a screenshot. By comparing the clock states in the resulting image, the G2G delay can be obtained.

For this setup, we utilized the Oak-1-max camera connected to three experimental devices: a Lenovo Legion laptop, an HP laptop, and a Jetson Orin board. In all cases, the Lenovo Legion laptop displayed a stopwatch from an online website while the Oak-1 live streamed the video on different browser windows. To standardize the procedure, the camera was always positioned over its original box on the laptop surface, between the keyboard and touchpad.

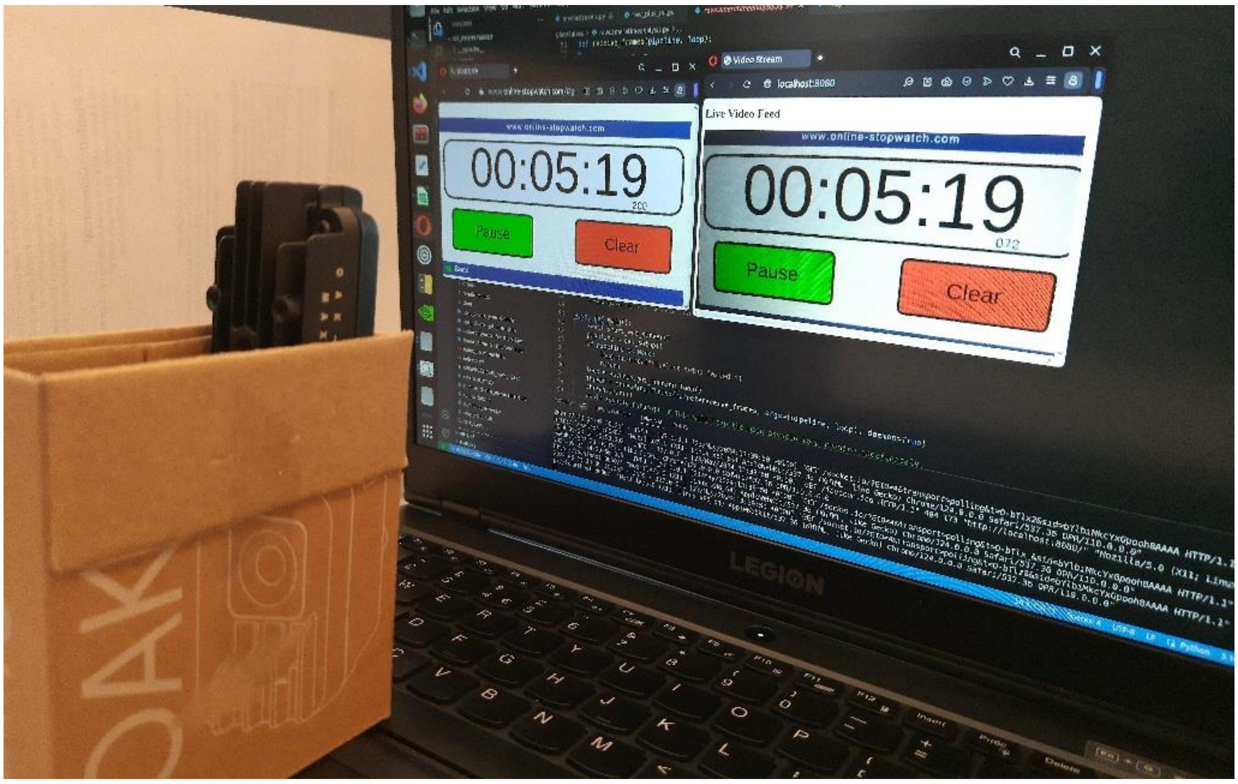


Figure 2: Glass-to-glass setup using Oak-1 and Laptop Lenovo.

3.13.1. G2G Procedure

To avoid manually calculating the delay by reading the numbers from the final screenshot, a semi-automatic post-process was developed. Initially, the reference clock (an online stopwatch) and the live stream windows were aligned on the screen. Based on their positions, a Python script was coded to take a screenshot of the specific area of interest every 3 seconds to avoid sequential delay.

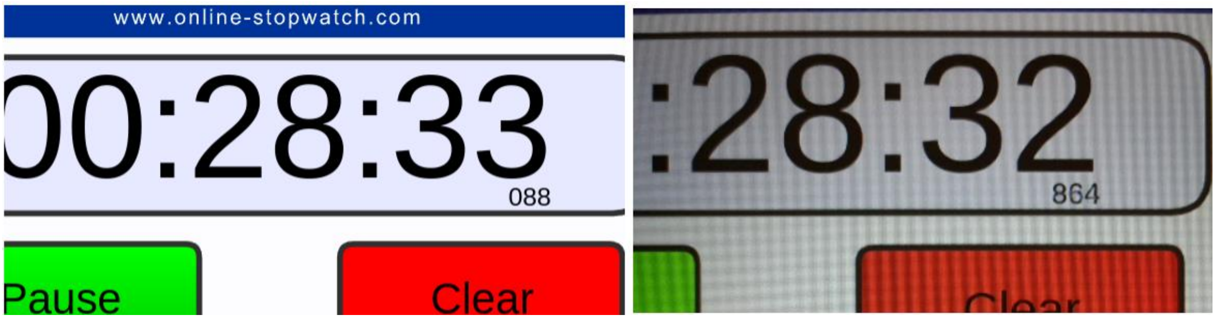
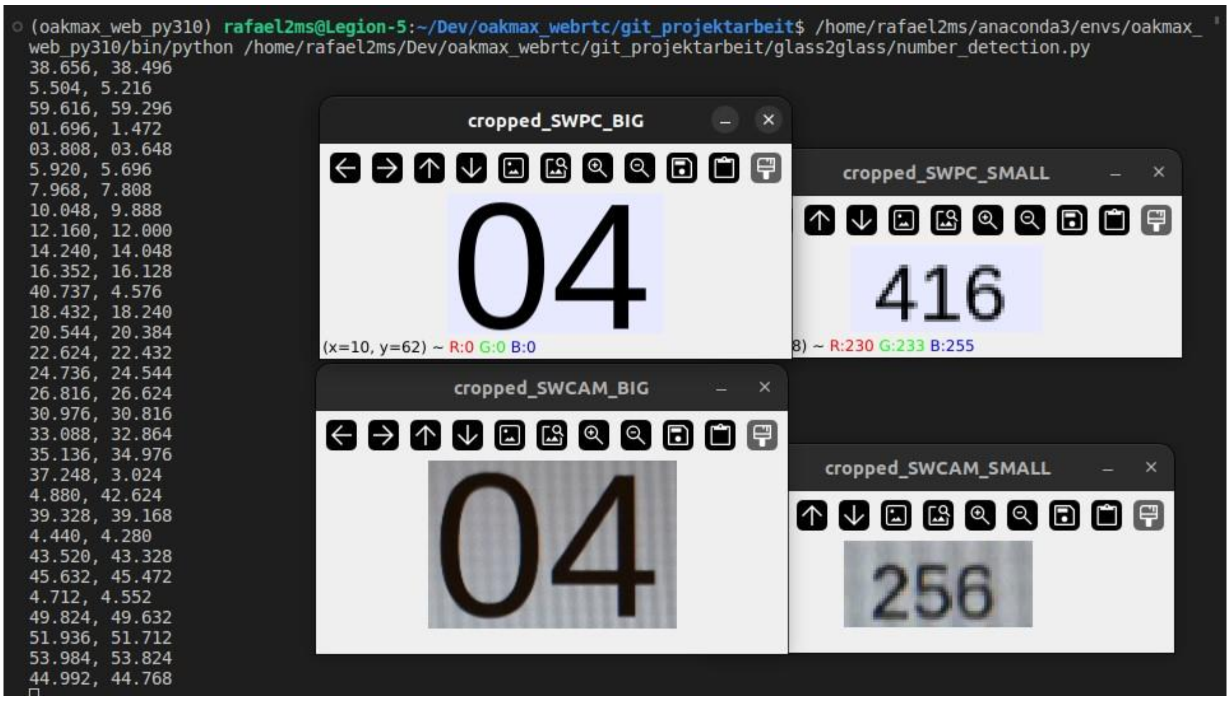


Figure 3: Cropped screenshot tool during Glass-to-glass measurement.

After 100 samples were collected, the script completed its process. The next step involved manually checking the quality of the screenshots, particularly focusing on how the milliseconds numbers were displayed by the livestream window. This was necessary because in about 20% of the measurements, there was a blurry transition of numbers due to the frame rate update by the screen and the camera delay in capturing the data.

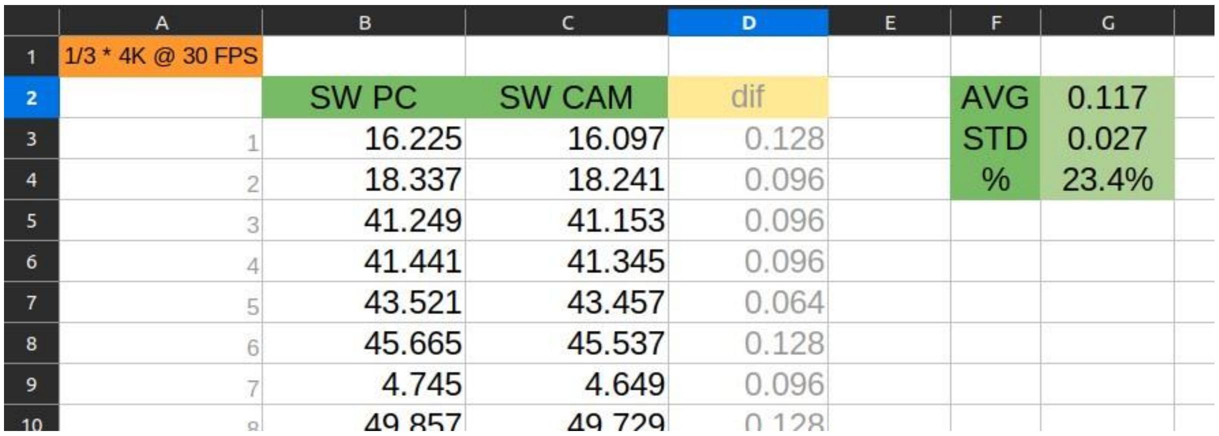
Next, these images were processed using another Python script that employed a machine learning model for number detection. The model, based on YOLO (You Only Look Once), was loaded for inference. The script then cropped the image to focus only on the area containing the numbers.



```
(oakmax_web_py310) rafael2ms@Legion-5:~/Dev/oakmax_webrtc/git_projektarbeit$ /home/rafael2ms/anaconda3/envs/oakmax_
web_py310/bin/python /home/rafael2ms/Dev/oakmax_webrtc/git_projektarbeit/glass2glass/number_detection.py
38.656, 38.496
5.504, 5.216
59.616, 59.296
01.696, 1.472
03.808, 03.648
5.920, 5.696
7.968, 7.808
10.048, 9.888
12.160, 12.000
14.240, 14.048
16.352, 16.128
40.737, 4.576
18.432, 18.240
20.544, 20.384
22.624, 22.432
24.736, 24.544
26.816, 26.624
30.976, 30.816
33.088, 32.864
35.136, 34.976
37.248, 3.024
4.880, 42.624
39.328, 39.168
4.440, 4.280
43.520, 43.328
45.632, 45.472
4.712, 4.552
49.824, 49.632
51.936, 51.712
53.984, 53.824
44.992, 44.768
```

Figure 4: Process of Screenshot Quality Check and Number Detection

All data collected was stored in a CSV file, followed by a visual inspection to validate the data by comparing the ML proposed numbers with the real ones displayed in the screenshot. Data analysis was performed using Microsoft Excel, focusing on the average and standard deviation among the samples.



	A	B	C	D	E	F	G
1	1/3 * 4K @ 30 FPS						
2		SW PC	SW CAM	dif		AVG	0.117
3	1	16.225	16.097	0.128		STD	0.027
4	2	18.337	18.241	0.096		%	23.4%
5	3	41.249	41.153	0.096			
6	4	41.441	41.345	0.096			
7	5	43.521	43.457	0.064			
8	6	45.665	45.537	0.128			
9	7	4.745	4.649	0.096			
10	8	49.857	49.729	0.128			

Figure 5: Screenshot of Excel Table Used for Manual Quality Check and Number Detection Process

This process was repeated to identify each intermediate latency present in the complete feedback-driven system, as described by the image below. By splitting the total E2E latency in subcomponents, the setup could be isolated to reproduce and measure each component Tn in the experiment.

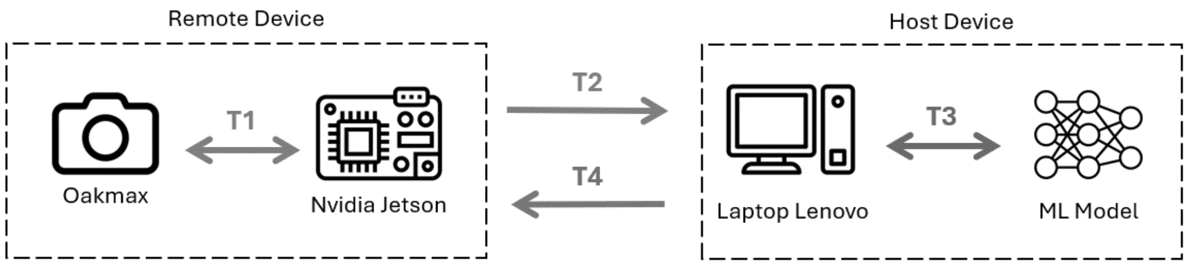


Figure 6: System's Latency subcomponents.

For calculating the end-to-end latency during crack segmentation (T3), a slight modification was made. An HTML script supported by JavaScript and CSS was developed to simultaneously display a stopwatch with milliseconds precision and a slideshow of crack images. This was necessary to trigger the segmentation model during the whole period of G2G measurement.

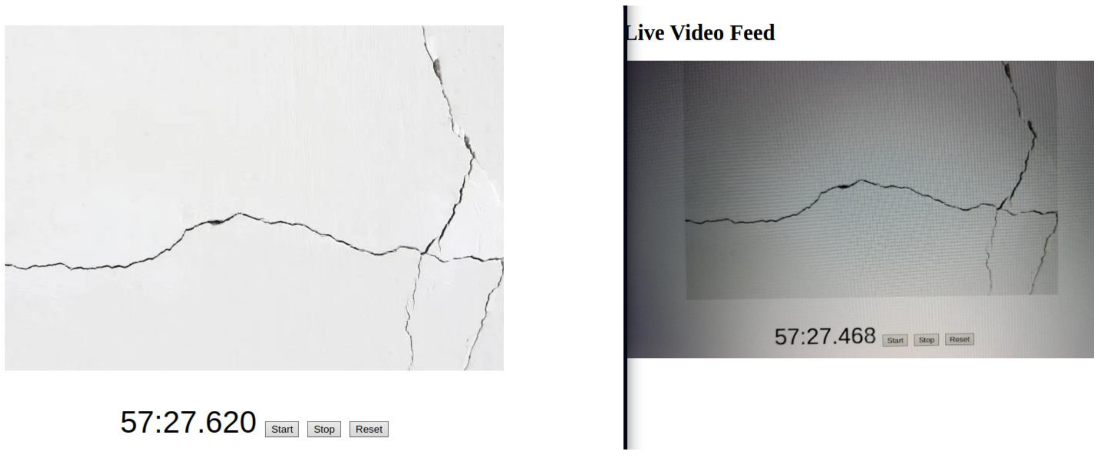


Figure 7: Example of G2G measurement using slide show with chronometer.

Notice that the streaming displayed on the right does not contain the segmentation mask. This was done to avoid adding latency to the process. To ensure the model was functioning correctly, the detection and segmentation layer, along with the directional algorithm from the PCA, were saved locally every 5 seconds. The dispersion in storage time and the isolated file visualization from the streaming minimized interference with the system.

The setup and procedure, while time-consuming and requiring significant human interaction, offered valuable insights into the latency components and variations. Despite the need for minor adjustments to frame the numbers correctly due to varying frame sizes, the cost-benefit of this preliminary study was deemed worthwhile, allowing for a comprehensive visualisation, and understanding of the latency involved in the video transmission process.

4. Results

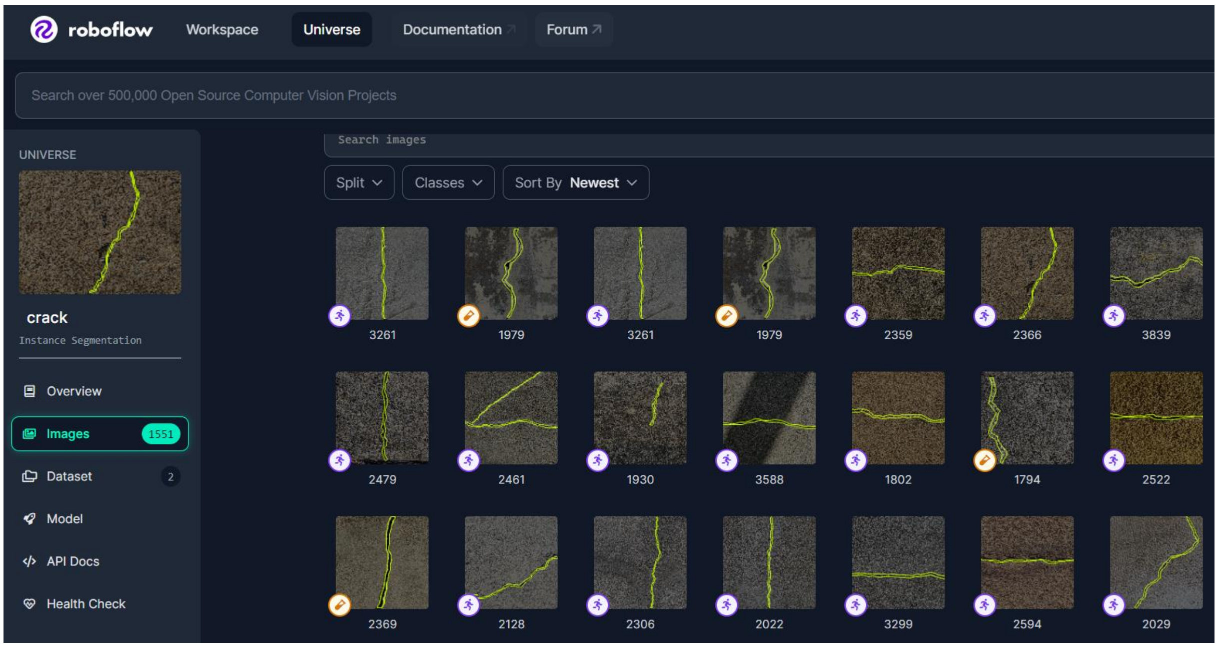
4.1. Crack Detection

This section presents the outcomes of the crack detection experiments, highlighting the datasets used and discussing a significant use case to demonstrate the practical application of the developed system.

4.1.1. Datasets Used

The crack detection model was trained and tested using the “Roboflow Computer Vision Dataset”, which includes a diverse range of images depicting various types of cracks.

- Training Data: Comprised 91% of the dataset, augmented with flips, rotations, shears, exposure adjustments, and noise additions to ensure robustness.
- Testing Data: Comprised 9% of the dataset, used to evaluate model performance.



4.1.2. Experimental Results

The experiments confirmed the system's capability to perform real-time crack detection effectively. The performance metrics, discussed in the methodology section, demonstrated high accuracy and efficiency.

- Object Detection and Segmentation: The model achieved high precision and recall rates, indicating accurate identification and segmentation of cracks.
- Latency and Performance: The system maintained low latency, essential for real-time applications, with computational efficiency aligning well with the hardware capabilities.

4.2. Max Pooling and PCA

To compare the efficiency provided by the application of Max Pooling before PCA, an experiment was conducted using a dataset consisting of 50 crack images, each with dimensions of 640x640 pixels. In the first scenario, the PCA method was applied directly to the original frames. In the second scenario, a Max Pooling downscaling with a factor of 16 was applied to the frames before performing PCA. For both cases, each frame was processed six times by the machine learning model, ensuring that any residual cache that could enhance performance was cleared.

The time measurement was performed using the Python "time" library by recording the time immediately before and after the execution of the PCA method. For each frame n, the average elapsed time (AET n) in PCA was calculated and compared between the two scenarios. The efficiency provided by Max Pooling to PCA (PEn^{PCA}) for each frame n is given by:

$$PE_n^{PCA} = \frac{\mathrm{AET}_n\ (\mathrm{PCA}_{640\mathrm{x}640})\ -\ \mathrm{AET}_n\ (\mathrm{PCA}_{40\mathrm{x}40})}{\mathrm{AET}_n\ (\mathrm{PCA}_{640\mathrm{x}640})} \times 100\%$$

Then, the average of this improvement across the 50 frames was calculated using the formula:

$$Average\ Improvement\ (PCA) = \frac{1}{50} * \sum_{n=1}^{50} PE_n^{PCA} = 76.9\%$$

Overall, executing PCA after downscaling the frame dimensions by a factor of 16 using Max Pooling resulted in a 76.9% reduction in time. Naturally, the time required for Max Pooling was also considered. To account for this, the same procedure was adopted, but this time comparing the average elapsed time between PCA alone (AETn) and the combination of Max Pooling plus PCA (PEn^{PCA+MP}):

$$PE_n^{PCA} = \frac{\mathrm{AET}_n\ (\mathrm{PCA}_{640\mathrm{x}640})\ -\ \mathrm{AET}_n\ (\mathrm{PCA} + \mathrm{MP}_{40\mathrm{x}40})}{\mathrm{AET}_n\ (\mathrm{PCA}_{640\mathrm{x}640})} \times 100\%$$

The average of the improvement among the 50 frames was calculated by:

$$Average\ Improvement\ (PCA + MP) = \frac{1}{50} * \sum_{n=1}^{50} PE_n^{PCA+MP} = 68.1\%$$

Hence, the application of Max Pooling before PCA provided a 68.1% faster calculation of the crack trend line, even considering the time spent on downscaling.

4.3. Results of Crack Segmentation Tracking

The images below illustrate the decision-making process described previously in the methodology over four real frames, arranged in two columns and four rows. The first two rows represent the scenario where the centre of the CF is already aligned with the CTL, while the last two rows depict the scenario where alignment is still necessary.

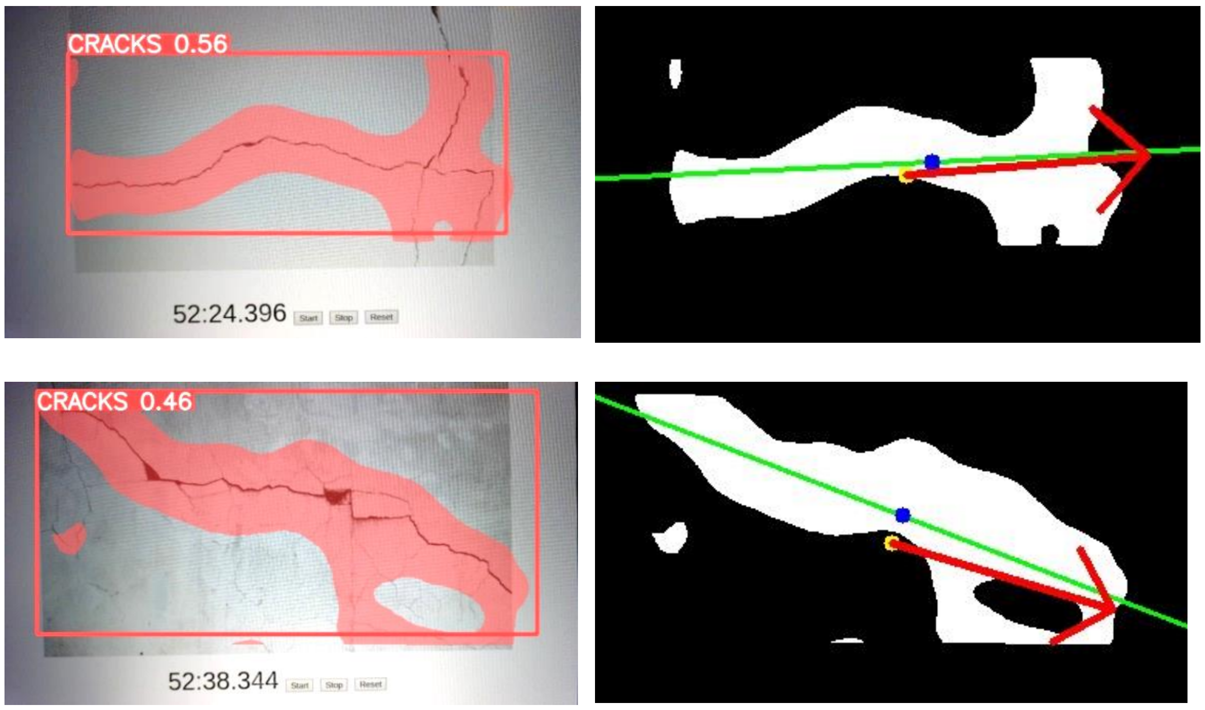


Figure 8: Set of frames which the crack is aligned and the direction algorithm pointing to the future trajectory.

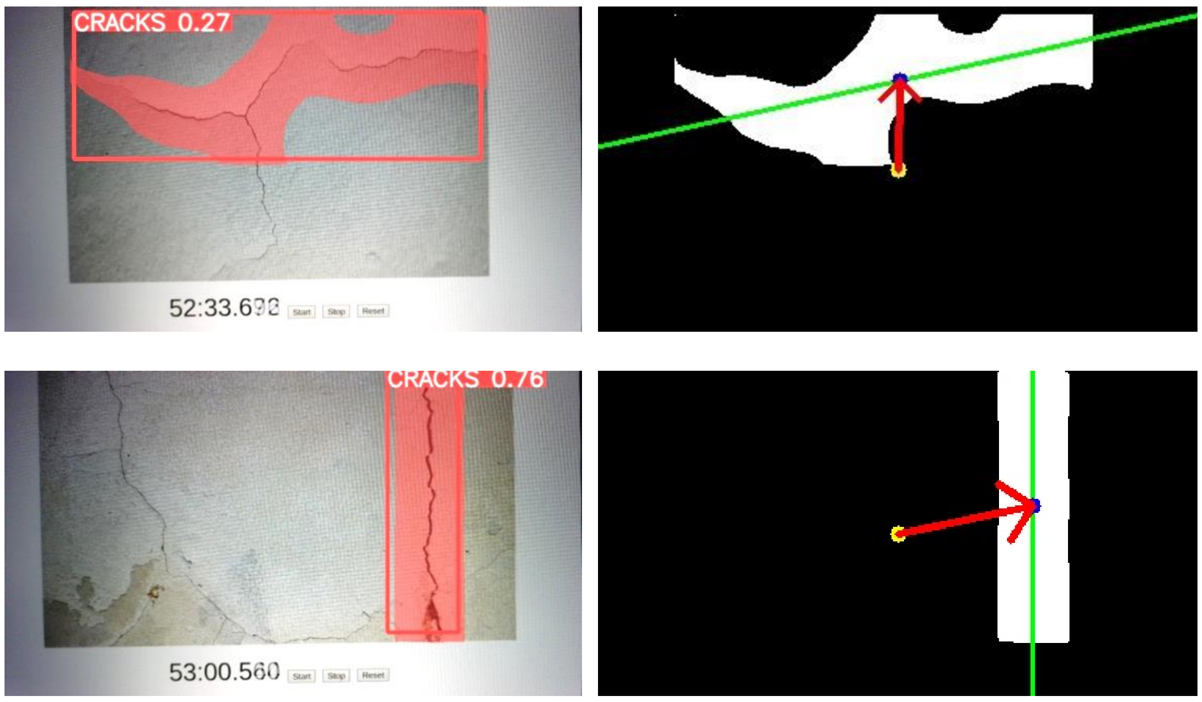


Figure 9: Set of frames which the direction algorithm points aim the alignment of the centre of the frame with the centre of the trend line.

In the left column, we can see the captured frames with the segmentation layer and the rectangular detection layer overlaid. The right column displays the post-processing frames, where white represents the segmentation mask and black indicates the non-detected area. The green line represents the trend provided by the PCA method. Additionally, the centre of the frame is marked in yellow, and the centre of the trend line is marked in blue. The direction taken, as determined by the previously mentioned algorithm, is shown by a red arrow.

It is important to address that the cases shown above were conducted using a reduction factor of 2 in the Max Pooling algorithm. This means that the frame dimensions were only halved before applying the Principal Component Analysis (PCA) algorithm. This approach was chosen to make the visualization of the entire procedure easier to understand, as it would be more demanding in a practical application.

For comparison purposes, using the last frame as an example, the images below contrast the PCA output from a reduction factor of 2 with that of a reduction factor of 16. This comparison highlights the impact of different down sampling levels on the PCA results, providing a clearer understanding of how frame reduction affects the overall performance and accuracy of the crack detection system.

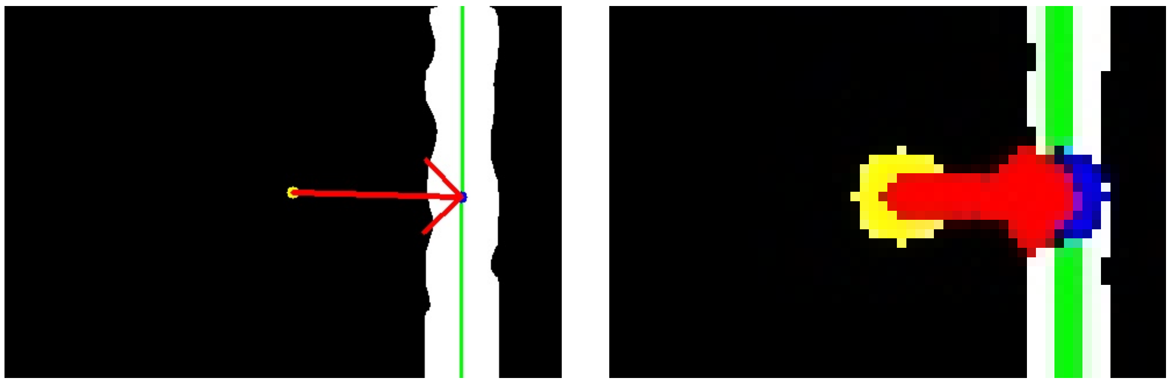


Figure 10: Conservation of navigation direction after downsizing through Max Pooling

4.4. End-to-End (E2E) Latency

The objective of this study was to ascertain the impact of network characteristics on end-to-end (E2E) latency from the perspective of the host device across all stages of this project. E2E latency is defined as the interval from the moment a frame is captured via the Software Development Kit (SDK) (instant T0) to the receipt of the directional command from the crack tracking algorithm (instant T5), with all data exchanges facilitated through WebRTC. To unravel the latency at various sub-steps in the process, additional timestamps were incorporated on the host device side (instants T1 to T4).

Consistency across the experiments was maintained by using identical scripts for all setups. Moreover, the Oak-1 camera was positioned consistently in each test, capturing the same view to ensure uniformity in the visual data collected. The sole variation involved the address of the signalling server, which was adjusted in response to changes in the network configuration. Additionally, to assist in latency measurements, three different frame sizes were tested at 30 FPS: 768x432 (S_1), 960x540 (S_2), and

1280x720 (S_3). The Precision Time Protocol (PTP) was implemented using a cross-over connection with an RJ45 cable, a critical step in synchronising the clocks of the devices involved.

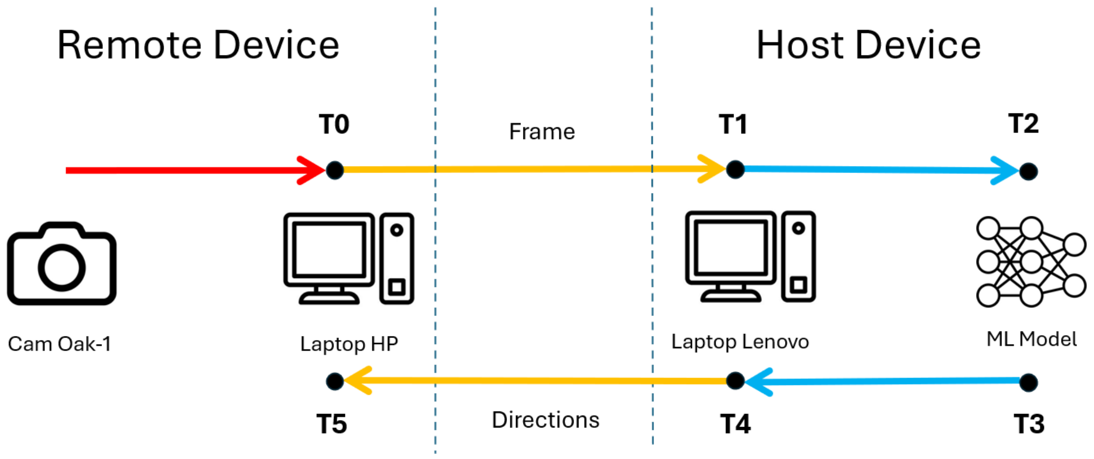


Figure 11: Instant of interest for E2E analysis of the feedback-driven decision system.

The technique that facilitated this experiment involved the intermittent insertion of a single 'real' frame amidst a continuous stream of 'white frames'. As the frames lack explicit identifiers, simply measuring time at different points during the transmission would not accurately determine which frame each measurement corresponded to. Moreover, reducing the frame rate to around 1 FPS would not realistically represent the WebRTC bit load and buffering.

Therefore, the proposed solution maintained a constant transmission rate of 30 FPS, sending a white image of the same size as the camera-captured frame, with a 'true' frame interspersed every three seconds—well beyond the latency observed in prior experiments. This approach ensured consistent traffic throughout the experiment for both transmission and reception, since using a white frame guaranteed that the machine learning output would remain constant, facilitating easy filtering. Additionally, maintaining a visible crack in front of the camera ensured that, given the model's high performance, detection was highly likely.

4.4.1. Setup 1: Localhost Environment

This setup involved only the Oak-1 and a Lenovo laptop, utilising the WebRTC protocol over “localhost:8080”. This configuration aimed to determine the system-intrinsic latency, isolated from external network influences.

	End-to-end Latency		
	S_1	S_2	S_3
AVG	71.354	73.072	82.404
STD	7.827	5.200	8.951
STD %	11%	7.1%	11%

Table 4.4-1: E2E Latency under localhost for three experimental frame sizes using Setup 1: Localhost Environment

Note: All delta values (ΔT0,1, ΔT1,2, etc.) and E2E (Sn) (sizes “S”) are in milliseconds (ms). Besides, weight values represent the proportional contribution of each delta time to the overall latency E2E, with a sum of 100%. They allow us to identify the most latency-influent tasks and possible bottlenecks.

- S_1 (768x432)

	$\Delta T_{0,1}$	$\Delta T_{1,2}$	$\Delta T_{2,3}$	$\Delta T_{3,4}$	$\Delta T_{4,5}$
AVG	41.893	0.022	29.003	0.011	0.569
STD	5.695	0.005	3.821	0.002	0.263
STD %	13.59%	22.75%	13.17%	22.75%	46.23%
Weight	58.71%	0.03%	40.65%	0.02%	0.80%

Table 4.4-2: Latency Measurements at Specific Intervals (ΔT0,1 to ΔT4,5) Within Our Feedback-Driven Decision System for a Frame Size of 768x432 using Setup 1: Localhost Environment.

- S_2 (960x540)

	$\Delta T_{0,1}$	$\Delta T_{1,2}$	$\Delta T_{2,3}$	$\Delta T_{3,4}$	$\Delta T_{4,5}$
AVG	42.307	0.023	29.924	0.011	0.736
STD	2.790	0.006	4.125	0.003	0.342
STD %	6.60%	27.72%	13.79%	27.72%	46.43%
Weight	57.90%	0.03%	40.95%	0.02%	1.01%

Table 4.4-3: Latency Measurements at Specific Intervals (ΔT0,1 to ΔT4,5) Within Our Feedback-Driven Decision System for a Frame Size of 960x540 using Setup 1: Localhost Environment.

- S_3 (1280x720)

	$\Delta T_{0,1}$	$\Delta T_{1,2}$	$\Delta T_{2,3}$	$\Delta T_{3,4}$	$\Delta T_{4,5}$
AVG	52.276	0.020	29.309	0.010	0.909
STD	5.887	0.003	5.496	0.001	0.272
STD %	11.26%	13.68%	18.75%	13.68%	29.97%
Weight	63.44%	0.02%	35.57%	0.01%	1.10%

Table 4.4-4: Latency Measurements at Specific Intervals (ΔT0,1 to ΔT4,5) Within Our Feedback-Driven Decision System for a Frame Size of 1280x720 using Setup 1: Localhost Environment.

4.4.2. Setup 2: LAN-Based Configuration

In this setup, both laptops were connected to the same Local Area Network (LAN), specifically 'ZHAW-Lowband'. The signalling server was hosted on the Lenovo laptop's IP address. Despite being on the same network, variations in service quality were observed, highlighting the inconsistency even within a controlled network environment.

	End-to-end Latency		
	S_1	S_2	S_3
AVG	122.299	144.879	221.323
STD	17.776	19.877	23.746
STD %	14.53%	13.72%	10.73%

Table 4.4-5: E2E Latency under localhost for three experimental frame sizes using Setup 2: LAN-Based Configuration

- S_1 (768x432)

	$\Delta T_{0,1}$	$\Delta T_{1,2}$	$\Delta T_{2,3}$	$\Delta T_{3,4}$	$\Delta T_{4,5}$
AVG	89.390	0.025	30.486	0.012	2.049
STD	17.945	0.005	5.022	0.003	0.514
STD %	20.07%	22.34%	16.47%	22.34%	25.09%
Weight	73.09%	0.02%	24.93%	0.01%	1.68%

Table 4.4-6: Latency Measurements at Specific Intervals (ΔT0,1 to ΔT4,5) Within Our Feedback-Driven Decision System for a Frame Size of 768x432 using Setup 2: LAN-Based Configuration.

- S_2 (960x540)

	$\Delta T_{0,1}$	$\Delta T_{1,2}$	$\Delta T_{2,3}$	$\Delta T_{3,4}$	$\Delta T_{4,5}$
AVG	105.939	0.024	36.003	0.012	0.736
STD	11.186	0.005	13.384	0.002	0.342
STD %	10.56%	20.48%	37.17%	20.48%	35.98%
Weight	73.12%	0.02%	24.85%	0.01%	1.81%

Table 4.4-7: Latency Measurements at Specific Intervals (ΔT0,1 to ΔT4,5) Within Our Feedback-Driven Decision System for a Frame Size of 960x540 using Setup 2: LAN-Based Configuration.

- S_3 (1280x720)

	$\Delta T_{0,1}$	$\Delta T_{1,2}$	$\Delta T_{2,3}$	$\Delta T_{3,4}$	$\Delta T_{4,5}$
AVG	174.166	0.030	44.317	0.015	3.276
STD	16.137	0.009	12.996	0.004	1.831
STD %	9.27%	29.03%	29.32%	29.03%	55.87%
Weight	78.52%	0.01%	19.98%	0.01%	1.48%

Table 4.4-8: Latency Measurements at Specific Intervals (ΔT0,1 to ΔT4,5) Within Our Feedback-Driven Decision System for a Frame Size of 1280x720 using Setup 2: LAN-Based Configuration.

4.4.3. Setup 3: Public Server Exposure via Local Tunnel

Utilising the "Localtunnel" tool, this setup involved exposing the local server on the Lenovo laptop to the internet, allowing an HP laptop to connect via a public URL using an LTE (4G) data connection. This arrangement tested the performance and stability of the system when interfaced with public internet infrastructure.

	End-to-end Latency		
	S_1	S_2	S_3
AVG	113.533	139.784	195.769
STD	11.292	12.918	18.148
STD %	9.95%	9.24%	9.27%

Table 4.4-9: E2E Latency under localhost for three experimental frame sizes using Setup 3: Public Server Exposure via Local Tunnel

- S1 (768x432)

	$\Delta T_{0,1}$	$\Delta T_{1,2}$	$\Delta T_{2,3}$	$\Delta T_{3,4}$	$\Delta T_{4,5}$
AVG	77.424	0.022	33.903	0.011	2.171
STD	8.359	0.005	6.113	0.003	1.232
STD %	10.80%	22.47%	18.03%	22.47%	56.77%
Weight	68.19%	0.02%	29.86%	0.01%	1.91%

Table 4.4-10: Latency Measurements at Specific Intervals (ΔT0,1 to ΔT4,5) Within Our Feedback-Driven Decision System for a Frame Size of 768x432 using Setup 3: Public Server Exposure via Local Tunnel.

- S2 (960x540)

	$\Delta T_{0,1}$	$\Delta T_{1,2}$	$\Delta T_{2,3}$	$\Delta T_{3,4}$	$\Delta T_{4,5}$
AVG	101.854	0.021	35.330	0.011	2.564
STD	10.042	0.004	8.263	0.002	1.056
STD %	9.86%	18.92%	23.39%	18.92%	41.18%
Weight	72.87%	0.02%	25.27%	0.01%	1.83%

Table 4.4-11: Latency Measurements at Specific Intervals (ΔT0,1 to ΔT4,5) Within Our Feedback-Driven Decision System for a Frame Size of 960x540 using Setup 3: Public Server Exposure via Local Tunnel

- S_3 (1280x720)

	$\Delta T_{0,1}$	$\Delta T_{1,2}$	$\Delta T_{2,3}$	$\Delta T_{3,4}$	$\Delta T_{4,5}$
AVG	158.329	0.022	34.952	0.011	2.455
STD	16.461	0.005	7.511	0.002	1.351
STD %	10.40%	20.52%	21.49%	20.52%	55.04%
Weight	80.88%	0.01%	17.85%	0.01%	1.25%

Table 4.4-12: Latency Measurements at Specific Intervals (ΔT0,1 to ΔT4,5) Within Our Feedback-Driven Decision System for a Frame Size of 1280x720 using Setup 3: Public Server Exposure via Local Tunnel.

4.5. Glass to Glass Results

The data presented in this section was gathered using the glass-to-glass measurement method, as detailed in the Methodology section. for all setups introduced in this section.

$$T\ total = T1 + T2 + T3 + T4$$

T1) Determine the latency on the visualization of raw camera image (without processing) caused by the combination of the camera's circuit, its SDK, and the webserver. The setup consisted in the Oak camera and the Lenovo laptop. For this measurements, five different resolutions were chosen, being displayed a fix frame rate of 30 fps.

Resolution	AVG LAT [ms]	STD [ms]	STD %
384x216	0.098	0.025	25.13%
768x432	0.098	0.027	27.55%
960x540	0.117	0.022	18.77%
1536x864	0.936	0.480	51.33%
1728x972	12.241	1.378	11.26%

T2) Next, using the same setup, the WebRTC protocol were added to the system, but running only locally. This would allow estimate the latency introduced by the protocol's implementation itself, without influence of internet. As for the first case, five different resolutions were chosen, and the frame rate stayed fix as 30fps.

Resolution	AVG LAT [ms]	STD [ms]	STD %
384x216	0.174	0.029	16.43%
768x432	0.189	0.031	16.59%
960x540	0.197	0.030	15.21%
1280x720	0.217	0.028	13.09%
1536x864	9.631	2.546	26.43%
1728x972	0.174	-	-

With both data, we can calculate the difference between them and estimate how many milliseconds were add on the system by the WebRTC structure, where the Difference in Averages is given by:

$$\Delta AVG = AVG2 - AVG1$$

And the Error Propagation, which stands by the standard deviation of the difference between two independent measurements, is calculated by the square root of the sum of the squares of the individual standard deviations.

$$\Delta STD = \sqrt{STD^2 + STD^2}$$

Resolution	ΔAVG [ms]	ΔSTD [ms]	STD %
384x216	0.076	0.038	50.00%
768x432	0.091	0.041	45.05%
960x540	0.080	0.037	46.25%
1536x864	8.695	2.590	29.79%

For the resolution of 1728x972, the system could not handle the streaming at 30 FPS and the transmission froze. In other hand, the resolution 384x216 stood of being the fastest transmission. Although it was fundamental to understand the relation between frame size and latency, its dimension was half of the trained by the ML model, what could lead to undesirable performance. Hence, both resolutions were discontinued for further experiments.

T3) Going to the most relative latency, a remote device was introduced to stream the oak-1 images through Wi-Fi via WebRTC. In a first moment, the Jetson Orin and then the HP laptop. Due to the inconstancy of the internet quality in different moments, a series of parameters of the Wi-Fi condition when the experiment occurred were gathered.

1) Jetson Orin @ “ZHAW” (5GHz)

Resolution	AVG LAT [ms]	STD [ms]	STD %
768x432	0.197	0.041	20.95%
960x540	0.208	0.045	21.51%
1280x720	0.235	0.054	23.18%

2) HP laptop @ “ZHAW-lowband” (2.4GHz)

Resolution	AVG LAT [ms]	STD [ms]	STD %
768x432	0.222	0.048	21.59%
960x540	0.269	0.043	15.98%
1280x720	0.359	0.057	15.95%

5. Discussion

5.1. Interpretation of Results

This project aimed to develop a real-time communication system using 5G to enable deep learning on the cloud for robotic applications. The research was structured around three main questions: the feasibility of real-time remote feedback-driven action systems using 5G, achieving bearable latency with machine learning processing, and defining "real-time" in the context of human needs for AI-driven robotic systems.

5.1.1. Real-Time Communication Using 5G

The results, particularly those outlined in Section 4.4, indicate that real-time communication is achievable even without 5G technology. The experiments demonstrated bilateral communication with latency comparable to the 200 ms benchmark often cited in remote surgery scenarios. While this project did not involve devices separated by great distances, it successfully validated the concept within a localized setup, such as a small city's bridge inspection. The use of WebRTC helped manage frame resolution, maintaining a smooth streaming experience. This highlights that machine learning models, unlike humans, do not require high-definition images to function effectively, thus mitigating the impact of reduced frame size on end-to-end (E2E) latency.

5.1.2. Enabling Deep Learning on the Cloud

Beyond achieving bilateral communication, the project successfully generated commands from single frames, responding in real time to the machine learning model. The crack segmentation model performed admirably, with training executed on the "Roboflow" database and subsequent inference on local hardware proving effective. The results, showcased in Section , highlight the model's precision, reinforcing its utility in practical applications. Furthermore, the crack tracking algorithm, designed for simplicity and speed, exceeded expectations by maintaining robust performance even after significant frame down sampling. The use of max pooling before PCA improved processing efficiency by 68%, demonstrating a clear benefit in preprocessing for cloud-based deep learning applications.

5.1.3. Applications in Robotics

Robotics applications often involve a variety of sensors and do not necessarily depend on high-resolution images. The project's methodology is versatile, evidenced by its application in environments where the robot operates at a speed of 1 m/s with a potential drift of around 25 cm upon detecting a trigger frame.

This finding is particularly relevant for tasks like wall crack tracking, where the robot's proximity to the wall mitigates the impact of latency, enhancing detection reliability despite viewing angle limitations.

5.2. Contextualization

Comparing these findings with existing literature underscores the project's contributions to real-time communication and machine learning integration in robotics. Previous studies have established the feasibility of single-way remote operations under specific latency conditions, but this project expands on those foundations by demonstrating effective bilateral communication and real-time decision-making driven by machine learning. The observed performance improvements align with existing research on the benefits of preprocessing techniques like max pooling in reducing computational load and enhancing model efficiency.

5.3. Implications

The practical implications of this research are significant, particularly for real-time infrastructure monitoring and other robotic applications requiring swift and accurate data processing. Theoretically, the project supports the growing body of evidence that preprocessing can substantially improve machine learning model performance, especially in resource-constrained environments. Methodologically, the approach of integrating cloud-based deep learning with localized preprocessing sets a precedent for future studies aiming to balance computational efficiency and real-time application needs.

5.4. Issues

5.4.1. Integration Dongle and Jetson Orin

One significant challenge encountered during this project was integrating the 5G dongle RM502-QAE with the Jetson Orin. The primary obstacle was the absence of dedicated USB and LTE network management drivers, such as “option” and “qmi_wwan”. The initial attempt to resolve this issue involved searching for these drivers in the repositories of the installed OS kernel's source version (5.10.120). Unfortunately, only newer versions were available, leading to the exploration of alternative solutions.

The next approach was to install a second kernel (5.10.192) alongside the existing one. This process proved to be complex, as it required careful handling to avoid conflicts between the two kernel versions, both of which had a “5.10” folder. Despite multiple attempts, each resulting in new errors or directory issues, the final decision was to restore the factory settings. Before proceeding with this, a data backup was performed using the Nvidia SDK Manager software to safeguard existing data.

Initially, the board was running Ubuntu 20.04, and the upgrade to Ubuntu 22.04, combined with the ARM64 architecture, made finding compatible solutions even more difficult. Restoring the factory settings seemed promising initially, but the process encountered a series of errors during the OS installation phase. Although some errors were resolved, the board ended up lacking essential drivers, including the wireless driver, which prevented further downloads.

As a last resort, due to the extensive time required to address this problem, many drivers were transferred via USB stick. However, this led to an ongoing cycle of resolving dependency conflicts, making it a laborious and time-consuming task. Therefore, the integration had to be suspended, and the project continued only with the HP laptop working as a remote device. This outcome, although unexpected, was welcome since one of the motivations of this project is off-board processing. This laptop would never have been able to work along ML models apart from the implementation of this technique.

5.5. Limitations

5.5.1. Crack Detection

Hafiz et al. highlight the significant challenges in developing image-based crack detection systems, primarily concerning the sourcing of images for datasets. Their comprehensive review reveals that approximately 70% of researchers opt to create their own datasets, employing a variety of imaging technologies including line cameras, robotic scanning, and smartphones. This approach allows for the collection of specific data needed to meet the unique demands of crack detection algorithms. Conversely, about 30% of researchers utilize pre-established crack datasets, such as Crack500 and GAPs384, which serve as standards for model testing and training due to their extensive libraries of crack images, facilitating broader application and feasibility in machine learning contexts [18].

Furthermore, Hafiz et al. discuss the domains in which these crack detection methods are applied, observing a predominant focus on infrastructure-related components like pavements, roads, and bridges. A significant portion of the research is dedicated to general civil structures, enabling these technologies to address a wide range of infrastructural elements. However, applications in detecting cracks in materials such as leather, steel, and components of nuclear power plants were less common, found only in a limited number of studies. This indicates a potential area for expansion, suggesting that future research could benefit from exploring crack detection methods applicable to a diverse array of materials beyond traditional civil infrastructure.

5.5.2. Glass to Glass measurements

Although Glass-to-Glass (G2G) measurement is a straightforward technique, its application in this project revealed several limitations. To begin with, the time required to collect data from the screenshot windows, even with the assistance of the number recognition model, was substantial. The frequency of blurred number captures further diminished the procedure's efficiency and reliability. Moreover, achieving highly standardized measurements required precise positioning for both the main screenshot and the smaller crops needed to extract the numbers. These crops had to be repeatedly checked, as even minor movements of the camera or windows could result in misalignment during post-processing.

Apart from the initial setup challenges, a significant and problematic outcome emerged during the final analysis of our WebRTC implementation for streaming video content. Due to the requirements for handling multiple concurrent operations within the WebRTC framework, the use of the OpenCV library was discontinued. Instead, a web server was implemented to stream images from the camera. Although this adjustment ensured the functionality of the system, it introduced a considerable performance bottleneck, which varied notably depending on the implementation approach.

In scenarios where the system utilized a web server combined with a local WebRTC server, the G2G latency measurements were substantially lower compared to the system that integrated direct visualization with the Machine Learning (ML) model. Intuitively, one would expect the addition of an ML model—which inherently adds more processing overhead—to result in higher latency. However, the observed performance discrepancy can primarily be attributed to the different methods of server integration and task management, specifically the use of asynchronous programming versus multi-threading.

In the implementation using the “asyncio” library, the web server and WebRTC components were managed within an asynchronous event loop. This method is highly efficient for I/O-bound tasks, such as network communication and handling web requests. The asynchronous approach allows the event loop to manage multiple operations by pausing and resuming tasks efficiently. This non-blocking behaviour is crucial in reducing waiting times between operations, hence potentially lowering the G2G latency.

Conversely, the second implementation utilized traditional threading to manage concurrent operations. While threading is effective for parallelizing CPU-bound tasks, it introduces complexity in synchronizing threads and can lead to increased overhead due to context switching and thread management. Moreover, threads in Python are limited by the Global Interpreter Lock (GIL), which can prevent truly concurrent execution of multiple threads. This could inadvertently increase the latency, particularly in a CPU-intensive application involving real-time video processing and ML computations.

Thus, the most plausible explanation for the observed latency differences lies in how concurrency was handled in each setup. The asynchronous setup with “asyncio” likely provided more streamlined and efficient management of I/O operations, leading to reduced latency. On the other hand, the threading model, while effective in certain contexts, was perhaps less suited to the demands of high-frequency, real-time video processing required by our system.

5.6. Opportunities for Enhancement and Development

5.6.1. Drone Application

Incorporating sophisticated technologies as suggested in [43] can significantly enhance the capabilities of this project, especially in terms of accuracy and efficiency for remote crack detection. The use of a drone equipped with advanced sensors and algorithms for on-board processing, as outlined in the study, presents an intriguing avenue for improvement.

Specifically, integrating a lightweight crack classification and segmentation algorithm alongside a high-precision crack edge detection system could dramatically refine the effectiveness of crack mapping. These algorithms, when combined with a depth map produced by a binocular camera, could provide a robust real-time decision-making tool for autonomous navigation. This setup would not only improve the precision in detecting and mapping crack geometries but also enhance the operational efficiency of the robotic system.

Moreover, the study proposes a novel approach that incorporates crack information with drone flight control, enabling automated path planning and ensuring stable flight. This method ensures that the drone can approach areas of interest—namely, regions with detected cracks—while maintaining stability, which is critical for high-resolution imaging and precise measurements under varied environmental conditions.

Adopting such technologies could elevate the project's capabilities by allowing for the extraction of maximum crack width under non-ideal conditions and achieving millimetre-level accuracy in measurements. This would be particularly beneficial for thorough inspections and maintenance strategies in infrastructure management, potentially making the system more adaptive and responsive to real-world operational demands.

5.6.2. Optimizing WebRTC Communication

The analysis of time distribution reveals that the majority of latency within the system is attributable to transmission, which depends heavily on network specifications and protocol structure. Enhancing WebRTC communication could be a significant step towards optimization. As previously discussed, this protocol has the capability to adjust parameters based on device constraints. Developing a solution that refines this management to reduce latency, without compromising segmentation performance, is essential for reliable improvements.

5.6.3. Network Role and Mobile Network Implementation

The network plays a pivotal role in system performance. Comparing the specifications of Wi-Fi and 5G throughout this project suggests that implementing mobile networks might yield better results. However, the enhancement of such technologies is not solely the responsibility of developers; it also depends on the broader implementation of technology across urban environments. Therefore, projects like this not only underscore the importance of advanced network technologies but also contribute to the momentum needed for their broader implementation.

5.6.4. Enhancing Frame Post-Processing

Although it constitutes a smaller portion of the overall latency, frame post-processing remains significantly impactful. During the coding phase, allocating mathematical operations to the GPU significantly accelerated the process. Implementing solutions like GPU acceleration or max pooling, which has been extensively discussed, could further reduce latency in this project phase.

5.6.5. Hardware and Setup Improvements

The setup itself also plays a crucial role. Results outlined in Section 4.5 illustrate superior performance of the Jetson board compared to the HP laptop. From the perspective of the host device, this performance differential was significant. Enhancing machine learning processing using an External GPU (eGPU) connected via a Thunderbolt could provide a substantial boost. Additionally, it's important to recognize that the machine learning model itself could be updated for faster processing. Tools like YOLO are continuously evolving, making it imperative to keep the segmentation models up to date.

6. Conclusion

As we conclude this thesis, it is evident that the integration of 5G technology for real-time communication in robotic applications via cloud-based deep learning presents a transformative potential. The findings from this research demonstrate that real-time communication, crucial for tasks such as remote surgery or infrastructure inspection, can be achieved with acceptable latency levels, even without the complete implementation of 5G. This project has successfully leveraged WebRTC for effective frame management and used machine learning algorithms to process data with impressive accuracy and speed. The utilisation of off-board processing to leverage cloud capabilities effectively addresses the computational limitations of local hardware, such as low-performance laptops, enhancing the overall system efficiency.

Moreover, the comparative analysis of network performances between Wi-Fi and 5G within this project's scope suggests that the broader implementation of 5G could significantly enhance the operational capabilities of robotic systems. The exploration of max pooling and GPU utilisation for reducing latency has shown that careful architectural considerations can substantially mitigate potential bottlenecks in data processing. These insights not only contribute to the academic field but also offer practical blueprints for future technological enhancements in real-time data transmission and processing.

Looking ahead, the path is set for continued innovation and enhancement of these technologies. While this study provides a foundation, it also highlights areas requiring further exploration, such as the integration of more advanced machine learning models and the expansion of 5G infrastructure. The responsibility now lies in the hands of future researchers and technologists to build upon this work, optimising and innovating with the aim of reducing latency and increasing the reliability of remote robotic operations. As for the future, your task is not to foresee it, but to enable it. — Antoine de Saint Exupery. This project has taken steps towards enabling a future where real-time robotic systems are more autonomous and efficient, laying the groundwork for the next stages of technological advancement in the field.

7. Reference list

[1] R. Dangi, P. Lalwani, G. Choudhary, I. You, and G. Pau, ‘Study and Investigation on 5G Technology: A Systematic Review’, *Sensors*, vol. 22, no. 1, Art. no. 1, Jan. 2022, doi: 10.3390/s22010026.

[2] K. Shafique, B. A. Khawaja, F. Sabir, S. Qazi, and M. Mustaqim, ‘Internet of Things (IoT) for Next-Generation Smart Systems: A Review of Current Challenges, Future Trends and Prospects for Emerging 5G-IoT Scenarios’, *IEEE Access*, vol. 8, pp. 23022–23040, 2020, doi: 10.1109/ACCESS.2020.2970118.

[3] ‘Special Report 03/22: 5G roll-out in the EU’, Accessed: May 24, 2024. [Online]. Available: https://www.eca.europa.eu/lists/ecadocuments/sr22_03/sr_security-5g-networks_en.pdf

[4] J. Marescaux and F. Rubino, ‘Transcontinental Robot-Assisted Remote Telesurgery, Feasibility and Potential Applications’, in *Teleophthalmology*, K. Yogesan, S. Kumar, L. Goldschmidt, and J. Cuadros, Eds., Springer Berlin Heidelberg, 2006, pp. 261–265. doi: 10.1007/3-540-33714-8_31.

[5] B. Zheng, Z. Janmohamed, and C. L. MacKenzie, ‘Reaction times and the decision-making process in endoscopic surgery’, *Surg. Endosc.*, vol. 17, no. 9, pp. 1475–1480, Sep. 2003, doi: 10.1007/s00464-002-8759-0.

[6] K. Pandav, A. G. Te, N. Tomer, S. S. Nair, and A. K. Tewari, ‘Leveraging 5G technology for robotic surgery and cancer care’, *Cancer Rep.*, vol. 5, no. 8, p. e1595, Mar. 2022, doi: 10.1002/cnr2.1595.

[7] M. Hu *et al.*, ‘Remote Animal Experiments of “Tumai” Surgical Robot Based on 5G Technology’. Apr. 12, 2023. doi: 10.21203/rs.3.rs-2692427/v1.

[8] J. Zheng *et al.*, ‘5G ultra-remote robot-assisted laparoscopic surgery in China’, *Surg. Endosc.*, vol. 34, no. 11, pp. 5172–5180, Nov. 2020, doi: 10.1007/s00464-020-07823-x.

[9] T. M. Ward *et al.*, ‘Computer vision in surgery’, *Surgery*, vol. 169, no. 5, pp. 1253–1256, May 2021, doi: 10.1016/j.surg.2020.10.039.

[10] Y. LeCun, Y. Bengio, and G. Hinton, ‘Deep learning’, *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.

[11] J. Donahue *et al.*, ‘Long-Term Recurrent Convolutional Networks for Visual Recognition and Description’, presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 2625–2634. Accessed: May 24, 2024. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2015/html/Donahue_Long-Term_Recurrent_Convolutional_2015_CVPR_paper.html

[12] I. Laina *et al.*, ‘Concurrent Segmentation and Localization for Tracking of Surgical Instruments’, in *Medical Image Computing and Computer-Assisted Intervention − MICCAI 2017*, vol. 10434, M. Descoteaux, L. Maier-Hein, A. Franz, P. Jannin, D. L. Collins, and S. Duchesne, Eds., in Lecture Notes in Computer Science, vol. 10434. , Cham: Springer International Publishing, 2017, pp. 664–672. doi: 10.1007/978-3-319-66185-8_75.

[13] M. W. Akram *et al.*, ‘CNN based automatic detection of photovoltaic cell defects in electroluminescence images’, *Energy*, vol. 189, p. 116319, Dec. 2019, doi: 10.1016/j.energy.2019.116319.

[14] A. Galdelli *et al.*, ‘A Novel Remote Visual Inspection System for Bridge Predictive Maintenance’, *Remote Sens.*, vol. 14, no. 9, Art. no. 9, Jan. 2022, doi: 10.3390/rs14092248.

[15] H. C. Garcia and J. R. Villalobos, ‘Automated Refinement of Automated Visual Inspection Algorithms’, *IEEE Trans. Autom. Sci. Eng.*, vol. 6, no. 3, pp. 514–524, Jul. 2009, doi: 10.1109/TASE.2009.2021354.

[16] T. Zhivkov, E. I. Sklar, D. Botting, and S. Pearson, ‘5G on the Farm: Evaluating Wireless Network Capabilities and Needs for Agricultural Robotics’, *Machines*, vol. 11, no. 12, Art. no. 12, Dec. 2023, doi: 10.3390/machines11121064.

[17] Z. M. Çınar, A. Abdussalam Nuhu, Q. Zeeshan, O. Korhan, M. Asmael, and B. Safaei, ‘Machine Learning in Predictive Maintenance towards Sustainable Smart Manufacturing in Industry 4.0’, *Sustainability*, vol. 12, no. 19, Art. no. 19, Jan. 2020, doi: 10.3390/su12198211.

[18] H. S. Munawar, A. W. A. Hammad, A. Haddad, C. A. P. Soares, and S. T. Waller, ‘Image-Based Crack Detection Methods: A Review’, *Infrastructures*, vol. 6, no. 8, Art. no. 8, Aug. 2021, doi: 10.3390/infrastructures6080115.

[19] S. Rathi, O. Mirajkar, S. Shukla, L. Deshmukh, and L. Dangare, ‘Advancing Crack Detection Using Deep Learning Solutions for Automated Inspection of Metallic Surfaces’, *Indian J. Inf. Sources Serv.*, vol. 14, no. 1, pp. 93–100, Mar. 2024, doi: 10.51983/ijiss-2024.14.1.4003.

[20] F. Ni, J. Zhang, and Z. Chen, ‘Pixel-level crack delineation in images with convolutional feature fusion’, *Struct. Control Health Monit.*, vol. 26, no. 1, p. e2286, 2019, doi: 10.1002/stc.2286.

[21] L. Zhang, F. Yang, Y. Daniel Zhang, and Y. J. Zhu, ‘Road crack detection using deep convolutional neural network’, in *2016 IEEE International Conference on Image Processing (ICIP)*, Sep. 2016, pp. 3708–3712. doi: 10.1109/ICIP.2016.7533052.

[22] H. Oliveira and P. Correia, ‘Automatic Road Crack Detection and Characterization’, *Intell. Transp. Syst. IEEE Trans. On*, vol. 14, pp. 155–168, Mar. 2013, doi: 10.1109/TITS.2012.2208630.

[23] V. Curic, A. Landström, M. Thurley, and C. Luengo Hendriks, ‘Adaptive mathematical morphology – A survey of the field’, *Pattern Recognit. Lett.*, vol. 47, pp. 18–28, Oct. 2014, doi: 10.1016/j.patrec.2014.02.022.

[24] F. Yang, L. Zhang, S. Yu, D. Prokhorov, X. Mei, and H. Ling, ‘Feature Pyramid and Hierarchical Boosting Network for Pavement Crack Detection’. arXiv, Jan. 24, 2019. doi: 10.48550/arXiv.1901.06340.

[25] Y. Ma and G. Guo, *Support Vector Machines Applications*. 2014, p. 302. doi: 10.1007/978-3-319-02300-7.

[26] R. Nagalla, P. Pothuganti, and D. Pawar, ‘Analyzing Gap Acceptance Behavior at Unsignalized Intersections Using Support Vector Machines, Decision Tree and Random Forests’, *Procedia Comput. Sci.*, vol. 109, pp. 474–481, Dec. 2017, doi: 10.1016/j.procs.2017.05.312.

[27] L. Wu, S. Mokhtari, A. Nazef, B. Nam, and H.-B. Yun, ‘Improvement of Crack-Detection Accuracy Using a Novel Crack Defragmentation Technique in Image-Based Road Assessment’, *J. Comput. Civ. Eng.*, vol. 30, no. 1, p. 04014118, Jan. 2016, doi: 10.1061/(ASCE)CP.1943-5487.0000451.

[28] O. Rainio, J. Teuho, and R. Klén, ‘Evaluation metrics and statistical tests for machine learning’, *Sci. Rep.*, vol. 14, no. 1, p. 6086, Mar. 2024, doi: 10.1038/s41598-024-56706-x.

[29] R. A. M. AlShehri and A. K. J. Saudagar, ‘Detecting Threats from Live Videos using Deep Learning Algorithms’, *Int. J. Adv. Comput. Sci. Appl.*, vol. 14, no. 11, 2023, doi: 10.14569/IJACSA.2023.0141166.

[30] S. X. Wei, ‘Data-Driven Safety-Critical Autonomy in Unknown, Unstructured, and Dynamic Environments’, phd, California Institute of Technology, 2024. doi: 10.7907/qpbp-0x81.

[31] N. Techasarntikul, H. Shimonishi, and M. Murata, ‘A Simulation of Energy Optimized Distributed Video Processing on 28 GHz Network’, in *2024 27th Conference on Innovation in Clouds, Internet and Networks (ICIN)*, Mar. 2024, pp. 265–271. doi: 10.1109/ICIN60470.2024.10494480.

[32] Y. Pan, T. O. Zander, and M. Klug, ‘Advancing passive BCIs: a feasibility study of two temporal derivative features and effect size-based feature selection in continuous online EEG-based machine error detection’, *Front. Neuroergonomics*, vol. 5, May 2024, doi: 10.3389/fnrgo.2024.1346791.

[33] B. Sredojev, D. Samardzija, and D. Posarac, ‘WebRTC technology overview and signaling solution design and implementation’, in *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, May 2015, pp. 1006–1009. doi: 10.1109/MIPRO.2015.7160422.

[34] ‘aiortc/aiortc’. aiortc, May 24, 2024. Accessed: May 24, 2024. [Online]. Available: https://github.com/aiortc/aiortc

[35] M. S. Khoirom, M. Sonia, B. Laikhuram, J. Laishram, and T. D. Singh, ‘Comparative Analysis of Python and Java for Beginners’, vol. 07, no. 08, 2020.

[36] ‘asyncio — Asynchronous I/O’, Python documentation. Accessed: May 24, 2024. [Online]. Available: https://docs.python.org/3/library/asyncio.html

[37] M. Greenacre, P. J. F. Groenen, T. Hastie, A. I. D’Enza, A. Markos, and E. Tuzhilina, ‘Principal component analysis’, *Nat. Rev. Methods Primer*, vol. 2, no. 1, pp. 1–21, Dec. 2022, doi: 10.1038/s43586-022-00184-w.

[38] D. H. Morais, ‘Wi-Fi 6 Overview’, in *5G NR, Wi-Fi 6, and Bluetooth LE 5: A Primer on Smartphone Wireless Technologies*, D. H. Morais, Ed., Cham: Springer Nature Switzerland, 2023, pp. 131–156. doi: 10.1007/978-3-031-33812-0_9.

[39] M. Hendaoui, ‘Integration of Wi-Fi, VLC, WiMAX, 4G, and 5G Technologies in Vehicular Ad Hoc Networks: A Comprehensive Review’, in *2023 7th International Symposium on Innovative Approaches in Smart Technologies (ISAS)*, Nov. 2023, pp. 1–6. doi: 10.1109/ISAS60782.2023.10391808.

[40] M. Lévesque and D. Tipper, ‘Improving the PTP synchronization accuracy under asymmetric delay conditions’, in *2015 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*, Oct. 2015, pp. 88–93. doi: 10.1109/ISPCS.2015.7324689.

[41] ‘NVIDIA Jetson Orin’, NVIDIA. Accessed: May 24, 2024. [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/

[42] C. Bachhuber and E. Steinbach, *A system for high precision glass-to-glass delay measurements in video communication*. 2016, p. 2136. doi: 10.1109/ICIP.2016.7532735.

[43] S. Meng, Z. Gao, Y. Zhou, B. He, and A. Djerrad, ‘Real-time automatic crack detection method based on drone’, *Comput.-Aided Civ. Infrastruct. Eng.*, vol. 38, no. 7, pp. 849–872, 2023, doi: 10.1111/mice.12918.

[44] ‘Sign in to Roboflow’. Accessed: May 24, 2024. [Online]. Available: https://app.roboflow.com/whatacrack

[45] ‘Roboflow’. Accessed: May 24, 2024. [Online]. Available: https://app.roboflow.com/whatacrack

Appendix A: Project Proposal

1. General Information

Course of study: EE Institute: IMS	Semester: FS24
Student 1: Rafael Monteiro Marques	Supervisor 1: Prof. Dr. Hans Wernher van de Venn Supervisor 2: Charith Munasinghe
External Advisor: Charith Munasinghe	
Credits: 6 ECTS Workload: approx. 96 Hours	Start: 19, February 2024 Deadline: 24, May 2024

1.1. Topic

Realtime communication using 5G to enable deep learning on the cloud for robotic applications.

1.2. Background and Motivation

The integration of Machine Learning models into various industries has witnessed a significant surge in recent years, driven largely by the flexibility and power of cloud computing platforms. These models, increasingly tailored to meet both personal and industrial requirements, often demand substantial computational resources, particularly in robotics where robust hardware typically equates to increased volume, weight, and power consumption.

Moreover, the off-board execution of deep learning models represents a significant leap forward in robotics, facilitating enhancements and customization of the control system via deep learning without requiring direct interaction with the robot. This advancement not only enables a single device to seamlessly switch between various tasks but also enhances its autonomous decision-making capabilities. The main requirement for these technologies is the development of a system capable of managing input, processing, and decision-making in a manner that meets real-time needs as perceived by humans, with latency low enough to accommodate specific application requirements.

The evolution of mobile networks marks a critical juncture in this context, especially with the advent of fifth generation (5G) technology. Characterized by its high data transfer capacity, the 5G network promises to revolutionize real-time video transmission with unparalleled stability. This feature, deemed essential for the advancement of remote surgeries and health-related applications [1], is also expected to be transformative in deploying ML models for object analysis within robotics. The widespread adoption of applications powered by this technology is drawing nearer, as evidenced by the European Union's ambitious plan to achieve comprehensive 5G coverage across all populated areas by 2030 [2].

Leveraging 5G's high-performance network capabilities for off-board ML model processing can significantly enhance remote operational efficiency in robotics. This approach addresses the logistical complexities and financial implications of transporting personnel and equipment for on-site evaluations.

Those issues are particularly acute in the maintenance of large-scale infrastructure such as bridges or buildings, for example. Such inspections and often require disrupting traffic, introducing both inconvenience and safety risks [3].

1.3. Project Goals and Objectives

1.3.1. Develop and Validate an ML Model for Crack Detection

a. Research and Selection: Identify and evaluate available pre-trained ML models for crack detection. Criteria for evaluation include accuracy, efficiency, and compatibility with the project's technical requirements.
b. Customization and Testing: Customise the selected model with a dataset specific to the project's focus area.

1.3.2. Assemble a Portable Inspection Device

a. Hardware Integration: Acquire and integrate a camera with a portable computer. Ensure the camera can be controlled via commands issued from the portable computer.
b. Wi-Fi Streaming Setup: Establish a reliable Wi-Fi connection for streaming images from the camera to a desktop environment.

1.3.3. Establish Callback Action Based on ML Output

a. Full-Duplex Communication Setup: Establish a system for sending decision commands from the desktop's ML evaluation back to the portable inspection device, concurrently with the live streaming transmission (as detailed in Objective 2, b).
b. Communication System Validation: Test the newly established full-duplex communication setup to ensure that decision commands are transmitted reliably and without disrupting the live streaming transmission.

1.3.4. Implement Off-Board ML Processing Over Wi-Fi

a. Desktop Model Execution: Run the validated ML model (from Objective 1) on a desktop using the images streamed from the portable inspection device (from Objective 2). This step validates the feasibility of off-board processing over Wi-Fi.
b. Performance Evaluation: Analyse the accuracy and efficiency of crack detection using the off-board ML model. Compare these results against initial on-board tests to assess the viability of off-board processing.

1.3.5. Upgrade to 5G for Enhanced Data Transmission

a. 5G Integration: Select and procure an appropriate 5G board and SIM card for the portable inspection device, ensuring compatibility with the existing hardware setup.
b. Network Configuration: Establish a 5G connection with the portable computer, configuring the system for optimal data transmission.
c. Protocol Testing: Conduct comparative tests of various 5G video streaming protocols to identify the most efficient protocol for real-time video transmission.

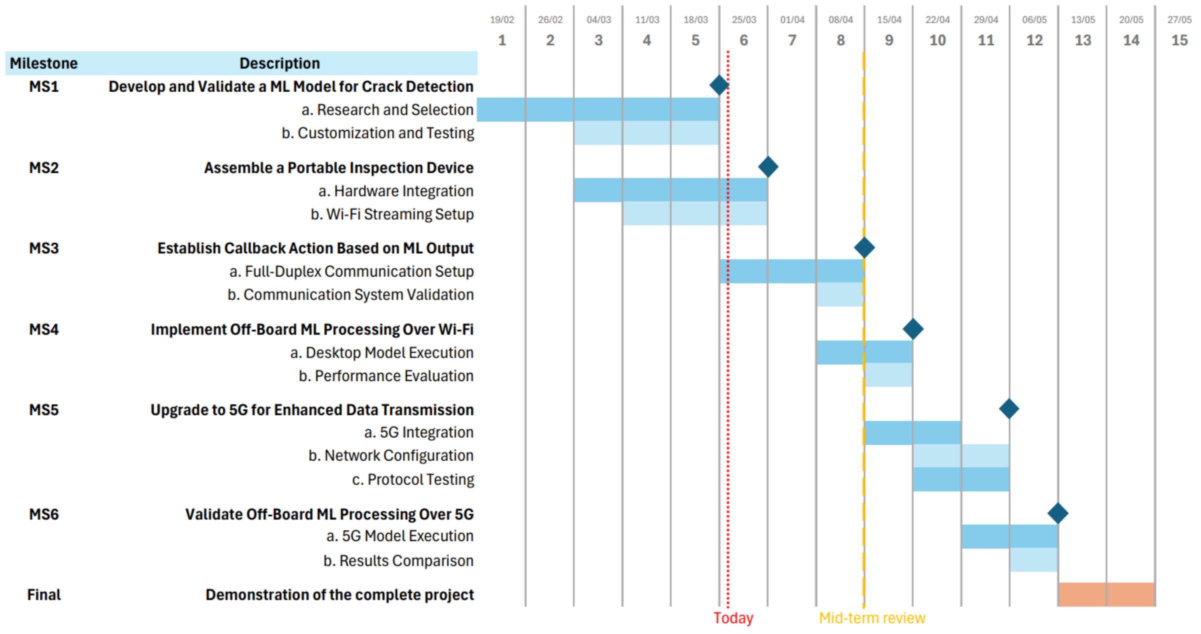
1.3.6. Validate Off-Board ML Processing Over 5G

a. 5G Model Execution: Rerun the ML model for crack detection using images streamed via the newly integrated 5G network. This step aims to evaluate the enhancements in transmission speed and model performance over 5G.
b. Final Assessment: Compare the results of off-board ML processing over 5G against the Wi-Fi-based setup. Focus on improvements in latency, video quality, and crack detection accuracy to validate the benefits of 5G integration.

1.4. Milestones and Deliverable

Milestone	Description
MS1	Develop and Validate an ML Model for Crack Detection
MS2	Assemble a Portable Inspection Device
MS3/Mid-term Review	Establish Callback Action Based on ML Output
MS4	Implement Off-Board ML Processing Over Wi-Fi
MS5	Upgrade to 5G for Enhanced Data Transmission
MS6	Validate Off-Board ML Processing Over 5G
Final Presentation	Demonstration of the complete project.

1.5. Project Planning and Management



a) Hardware Utilization: The project will leverage the advanced robotic hardware and supporting equipment available through the Robotic team at the Institute of Mechatronics Systems, ZHAW. This includes robotic platforms equipped with camera systems for real-time data capture.

b) Field Testing: Field tests will be conducted at selected ZHAW installations or relevant real-world structures to evaluate the system's performance in actual operating conditions. These tests aim to identify potential challenges and validate the system's effectiveness in real-time crack detection and structural health monitoring.

c) Testing in Controlled Environment: The selected protocols will be tested within ZHAW's facilities, utilizing the available 5G module, data network, and processing units. Tests will simulate various operational scenarios to assess each protocol's performance in terms of data transmission speed, latency, and overall Quality of Service (QoS).

d) AI Model Selection: If a reliable pre-trained AI model for crack detection is available, it will be integrated into the project. The selection will be based on the model's accuracy, efficiency, and adaptability to different structural conditions.

e) Evaluation and Validation: The project will evaluate the overall system's performance, focusing on the efficiency and reliability of the robot-control unit communication over a 5G network and the accuracy of the AI-powered crack detection.

1.6. Project Reports and Outcomes

- Reports on diploma thesis
  - Final report on the diploma thesis in two versions. (bounded-paper and electronic)
  - A project summary in poster form.
- A presentation on the project and demonstration of the outcome
  - The presentation as ppt/pdf documentation
  - Presentation to the panel including supervisors and other interested parties.

1.7. Frequent progress reports and agreement of timely feedback

Progress reports will be made every two weeks, in the form of a short meeting, presenting the work that was accomplished in the past two weeks, and defining the work for the next two weeks.

1.8. Commitment of good scientific practice

I hereby declare that I adhere to the principles and procedures of integrity in scientific research, laid down in [4].

Signature: [illegible]

Rafael Monteiro Marques

1.9. References

[1] The reuse policy of the European Court of Auditors (ECA) is implemented by Decision of the European Court of Auditors No 6-2019 on the open data policy and the reuse of documents. (link)

[2] Scalable Virtual Network Video-Optimizer for Adaptive Real-Time Video Transmission in 5G Networks" (2020) by P. Salva-Garcia, JM Alcaraz-Calero, et al.

[3] Applications of Unmanned Aerial Vehicle (UAV) in Road Safety, Traffic, and Highway Infrastructure Management: Recent Advances and Challenges (2020) by F. Outay, H.A. Mengash, et al.

[4] Bossi, E. et al., Integrity in scientific research - Principles and procedures. Swiss Academies of Arts and Sciences Hirschengraben 11, Postfach 8160, 3001 Bern Tel. 031 313 14 40, Fax 031 313 14 50 www.swiss-academies.ch, info@akademien-schweiz.ch © 2008.