TECHNISCHE UNIVERSITÄT
CHEMNITZ

Fakultät für Informatik

CSR-26-06

Simultaneous Video Streaming and Recording with Geotagging in Autonomous UAV

Silpa Geetha Harshakumar · Batbayar Battseren · Wolfram Hardt

Mai 2026

Chemnitzer Informatik-Berichte

TECHNISCHE UNIVERSITÄT
CHEMNITZ

Simultaneous Video Streaming and Recording with Geotagging in Autonomous UAV

Master Thesis

Submitted in Fulfilment of the
Requirements for the Academic Degree
M.Sc.

Dept. of Computer Science
Chair of Computer Engineering

Submitted by: Silpa Geetha Harshakumar
Student ID: 714848
Date: 05.02.2026

Supervising tutors:
Prof. Dr. Dr. h. c. Wolfram Hardt
Dr. Batbayar Battseren

Abstract

Unmanned Aerial Vehicles (UAVs) are currently becoming dependent on real-time video to be used in activities like surveillance, inspection, monitoring and situational awareness. These applications put a severe constraint on the low-latency live video delivery, consistency of onboard recording, and uniform correlation of visual data and telemetry data. Traditional UAV video systems usually provide streaming and recording as two different pipelines, thus resulting in poor use of resources and difficulty in ensuring synchronization on embedded systems. This thesis introduces a design and analysis of an integrated video processing system in UAVs that will provide them with the ability to stream video in real-time and record videos on command within the same unified multimedia pipeline. It is built on a NVIDIA Jetson platform on an NVIDIA GStreamer platform and with camera capture, telemetry and timestamp overlay, hardware-accelerated H.264 encoding as well as a branch of the pipeline with a tee element. Live video is transmitted to a Ground Control Station in a browser based application via the Janus Gateway via WebRTC and onboard recording can be enabled and disabled without affecting the live video. In order to have a steady geotagging process, timestamps and telemetry information is added to the video frames directly, both streamed and recorded outputs can retain the same visual metadata. Functional verification and quantitative analysis of performance are the means of evaluation of the system. The GStreamer latency tracer is used to measure internal pipeline latency and indicates low and predictable processing delay in pipelines (streaming and recording). The end-to-end frame continuity analysis makes sure the video delivery is stable with zero frame drops in the steady-state mode. Frame rate analysis indicates that performance is stable and on par with real time requirements and the amount of resources used is within the reach of the embedded platform. The findings confirm that a single and hardware-accelerated multimedia pipeline could be effectively used in supporting real-time video streaming, onboard recording and telemetry overlay on the resource-constrained UAV systems. The given architecture offers a scalable and robust architecture to UAV video applications with low latency, consistency of synchronization, and flexibility.

Keywords: Unmanned Aerial Vehicles, Real-Time Video Streaming, Geotagging, GStreamer, Jetson Nano

Acknowledgment

Thank you to all those who helped and guided me throughout my thesis journey. Their support, encouragement, and guidance have been invaluable to the successful completion of this work. First, I would like to sincerely thank Prof. Dr. W. Hardt, Chair of Computer Engineering at Technische Universität Chemnitz, for allowing me to carry out this thesis under his supervision. Without his continuous support, guidance, and academic insight, this research would not have been possible.

I would also like to express my deep gratitude to Dr. Batbayar Battseren for his constant encouragement, technical guidance, and valuable feedback throughout the course of this thesis. His suggestions and discussions played a significant role in keeping the work focused and on track, especially during challenging phases of the research.

I would like to thank my husband, Ambadi Mohandas Ajitha and my mother, Geetha S, together for their immense support and understanding throughout this journey. Their unwavering belief in me, emotional support, and constant encouragement gave me the strength and motivation to persevere and complete this work successfully. Finally, I would like to thank my friends, colleagues, and all those who contributed directly or indirectly to this thesis for their support and understanding. I am extremely grateful to everyone who accompanied and encouraged me throughout this academic journey.

Contents

List of Figures

List of Tables

List of Abbreviations

UAV Unmanned Aerial Vehicle

GCS Ground Control Station

FPS Frames Per Second

IMU Inertial Measurement Unit

GPS Global Positioning System

MAL MAVLink Abstraction Layer

MAVLink Micro Air Vehicle Link

V4L2 Video4Linux2

MJPEG Motion JPEG

NVENC NVIDIA Video Encoder

RTP Real-time Transport Protoco

UDP User Datagram Protocol

WebRTC Web Real-Time Communication

ICE Interactive Connectivity Establishment

DTLS Datagram Transport Layer Security

SRTP Secure Real-time Transport Protocol

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

JSON JavaScript Object Notation

MP4 MPEG-4 Part 14

MKV Matroska Video

HTML HyperText Markup Language

RTP/UDP Real-time Transport Protocol over UDP

GST GStreamer

JS JavaScript

SSE Server-Sent Events

1 Introduction

Unmanned Aerial Vehicles (UAVs) have indeed revolutionized a broad scope of civilian and industrial activities. UAVs' ability to access difficult or inaccessible areas has been a major advantage. As drone technology continues to advance, the need for intelligence and sophisticated sensing capabilities also continues to grow beyond merely navigating the drone. Among the most critical sensing capabilities of UAVs is video sensing. A video log collected from UAVs accumulates a plethora of information, enabling the user to assess the situation, confirm objectives, and respond to any changes in the environment in real time. The video stream also possesses a plethora of contextual information and situational cues that might otherwise be difficult to interpret from other sensing data, thereby becoming a critical component of UAVs, particularly when a human in the loop or a human judgment becomes critical [19].

Embedded computing plays a huge role in the processing of video data in UAVs. The older UAVs used relatively simpler camera systems that streamed the video straight down to the ground in its raw form, possibly compressed a little, and there was virtually no processing on the ground. The newer UAVs, on the other hand, are becoming more harnessed with a computer embedded within that can run a multimedia pipeline with on-board capability, thus reducing the dependency on processing on the ground [47].

However, there are a number of technical issues that are related to the implementation of real-time video processing for these UAV platforms. The UAV companion computer has limited computation, power, as well as cooling capabilities. On the other hand, video processing activities such as decoding, conversion, rendering, encoding, as well as network transmission are not only computation motor but also transmission time-sensitive. With a need to establish a system that will be able to handle real-time requirements while at the same time ensuring that it is reliable in operation, it is important that a selection of levels be carefully considered [4].

Along with the video, UAVs also generate copious amounts of telemetry data such as navigative data, sensor data, and system health data parameters. Telemetry data may also provide vital spatial and temporal background for the video, which may prove to be of considerable importance in the use of the video when location, altitude, or time stamping of the video is of primary importance. For example, in an inspection or disaster relief situation, the time and location stamp of the video frame may be of equal importance to the actual video. Although telemetry data is

of primary importance, the same can also be treated as a separate entity of data, which may or may not be stored or transmitted on the same channels as the video. This may prove to be a drawback in the effective and easier handling of the UAVs for the purpose of real-time analysis and post-mission analysis due to the need to incorporate additional synchronizing and special tools for the purpose of mapping the video and telemetry together. When the video is transmitted live and also stored on the UAV, the problem becomes even more complex because the synchronizing of the outputs cannot be maintained easily.

One major challenge associated with UAV video systems is that there is often a coexistence between video recording and video streaming, and such coexistence is often in a way that leads to competition for attention. There is a need for low latency in video streaming to ensure real-time awareness at the Ground Control Station, whereas there is a need for reliable video recording for post-flight analysis. However, most video systems are designed to focus on one goal at the expense of the other or use two separate paths with similar processes, making them inefficient for UAVs.

Aside from this, the relationship between operators and UAV video systems has evolved to a more web-centric approach, with an emphasis on simplicity and user-friendliness. GCS now runs in a browser, so there is no need to install special software to view video or send control commands. However, making these simple, lightweight video and control interfaces work with low-latency video feeds, as well as on-board recording and control, presents new challenges [10].

In these realities, there is clearly a need for a single UAV video system that can do real-time video streaming, on-demand video recording, and synchronized telemetry integration without sacrificing any of these capabilities due to constraints imposed by embedded systems. Such a system should be balanced between performance, flexibility, robustness, and usability in order for it to be expandable in the future.

This thesis will answer these questions with the development and application of a video processing framework, one that is modeled after the process flow of an autonomous UAV. Hardware-accelerated multimedia processing, modularity, and network/web technologies will be used to achieve real-time video streaming, as well as the capability to record video and even geotag it. The sections that follow will provide more background information to assist with the development of the problem statement and the study's objectives.

1.1 Background and Context

Unmanned Aerial Vehicles, also known as UAVs, began as a small-scale research project but has since grown into an integral part of daily life, used in a variety of

civilian and industrial settings. Its ability to maneuver and operate in 3D spaces, reach areas that are difficult to access, and act as a sensor has led to its adoption for tasks like monitoring, disasters, search and rescue, environmental tracking, farming, and surveillance. As UAV technology continues to advance, the need for more autonomous, intelligent, and advanced data handling is also increasing [16].

Video capturing and streaming is at the core of most UAV flights today. Viewing video provides a clear and intuitive understanding of the environment, which is difficult to achieve through other means of sensing, as they do not provide an understanding of how things are arranged and look and move in an intuitive fashion. This is why video streaming is such an essential part of UAVs today, particularly when there is a human involved [50].

With the continued evolution of video processing, the UAV systems produce massive amounts of telemetry data, which is extracted from navigation sensors, inertial measurement units, and system monitors. The aerial video becomes relevant when placed within the context of geographic coordinates, altitude, orientation, and time, among other factors. For example, in inspection and disaster response missions, having the exact location and time of every frame of the video is important.

This is particularly true with regard to the telemetry, as it is often treated as a separate entity from the video. For instance, most systems have the telemetry recorded separately, and this means that there is a need to match the video with the corresponding telemetry. As the complexity of the system is increasing, it is becoming more difficult to keep the outputs of the system in sync and consistency. The fact that the video is often streamed and a local copy is also recorded on the vessel complicates the situation.

Another part of the context is the dual requirement for live video streaming and on-board video recording. Live video streaming is focused on low latency, enabling video to be sent to the Ground Control Station quickly in order to react to what is happening in real time. On-board video recording, on the other hand, is focused on high reliability, making sure that the video is not lost when there is no or poor network availability. However, attempting to fulfill both requirements with a single solution is not straightforward, especially on resource-constrained platforms.

On an operational level, web-based GCS are also becoming more popular in the modern UAV systems. Platform-independent access: Browser interfaces make it easier to access UAV video systems by using common technologies, particularly since they are platform independent. Reduced deployment complexity: Browser-based interfaces enable the operator to interact with the UAV video systems with common technologies. The onboard processing, media servers and client-side technologies have to work closely to integrate real time video streaming, recording control and telemetry visualization with such interfaces. It is within this larger context that a

definite moving trend towards unified and modular UAV video architectures that are capable of efficiently managing the multiflow of data and which remain robust and flexible have emerged. These architectures should combine video processing, telemetry processing, and network streaming in a coherent way that does not redundantly perform computations nor give different behaviour to live and recorded results.

This is the background leading to the work in this thesis. Through the application of contemporary embedded computing platforms, hardware-based multimedia systems, and uniform networking systems, the proposed system will meet the difficulties involved in the concurrent streaming of video and recording of video with in-built geotagging. The subsequent paragraphs elaborate further on the underlying motivations, problem formulation and the goals between which the research will be directed.

1.2 Motivation

Real-time video is an important facility in UAV-based applications like surveillance, disaster management and defense whereby operational decision making is directly backed by timely visual information. Live video streams, in such a case, allow the operators to evaluate the changing conditions, react to threats and organize the actions effectively. Real-time information streaming to the Ground Control Station is thus necessary to have a situational awareness when executing missions.

Besides the real-time monitoring, post-mission analysis is another need of high importance. Aerial video that is recorded is common to watch the flight path and assess the outcome of the mission and assist in a more in-depth analysis once the mission has been completed. Video data that has been annotated with geotagging data, such as timestamps, geographic position, inertial metrics that allows more powerful analysis tools, including GIS-based visualization, synchronized replay, and precise spatial referencing. Metadata-containing recorded video is also verifiable evidence in security and law-enforcement uses, so the data integrity and contextual completeness is very important.

Nonetheless, not all current UAV video systems have been engineered to promote real-time streaming and onboard recording in such a way that it can proceed in one unified and cost-effective form. Streaming and recording are commonly thought of as dissimilar functionalities in that the processing phases are repeated, and embedded computing resources are ineffectively utilized. Moreover, telemetry data is often processed as another stream of data, and more other data synchronization processes must be performed in the course of post-processing, making the system more complex.

This is especially important to embedded UAV systems like the NVIDIA Jetson Nano, where computation resource, power consumption and thermal limits need to be highly considered. The use of several pipelines with the independence or the use of ground-side post-processing is not best suited to these platforms. Hence there is a definite requirement of an integrated video processing platform to enable sustained real time streaming, on-demand onboard recording and synchronized geotagging in a unitary, effective architecture.

This thesis is driven by the fact that such issues need to be addressed through the design and implementation of a system capable of supporting low-latency live streaming, selective onboard recording, and embedded telemetry overlays on embedded UAV platforms. This kind of system improves real-time decision-making on the flight and makes sure that the recorded information could be understood and interpreted in an immediate way and could be utilized in the post-mission analysis, visualization, and storage of evidence.

1.3 Problem Statement

Although there is a great improvement in UAV platform, embedded computer and multimedia system, none of these has a combination with a solution that is efficient to support real time video streaming, on board recording and synchronized geotagging with in the same system. Current UAV video systems tend to meet these needs in isolation which leads to fragmented architectures that are hard to scale, maintain and run on resource limited platforms.

An underlying issue is that streaming and recording pipelines are separated. In many UAV systems, live streaming and onboard recording are separate processes that both have a decoding, processing, and encoding process. This method results in wastes of computing, power, and complicates the system, which does not fit the embedded companion computers like the NVIDIA Jetson Nano. This consequently results in low levels of system stability and performance in real-time when run continuously.

The other problem is related to the inflexibility in terms of recording in the case of live streaming. In most current applications, recording has to be set on permanently or triggered prior to the commencement of the mission. This inflexible approach restricts flexibility of operations and causes an inefficient utilization of onboard storage because non-critical parts of the flight are needlessly documented. The fact that real-world missions on UAV video systems cannot be effectively used dynamically due to the inability to start and stop recording without interfering with the live video stream.

Telemetry and geotagging data also makes the issue more complicated. Time, position and orientation Telemetry information can be stored independently of video, or sent via other channels of communication. The separation adds a synchronization problem and needs further post-processing so as to match telemetry with the video recorded. When used in a more surveillance, inspection and law enforcement context, this absence of a direct connection lowers the immediacy and reliability of video-based evidence and analysis.

System integration-wise, there is also no cohesive, operator-friendly interfaces that can join live video visualization and recording control and telemetry awareness. Proprietary Ground Control Station software or custom client applications make deployments more complicated and reduce their accessibility. Short-latency streaming technologies and embedded video pipelines need to be carefully integrated with web-based interfaces to be a promising alternative.

Combined, these problems demonstrate the lack of a single, resource-efficient UAV video system that allows unlimited low-latency video streaming, on-demand, onboard recording and synchronized, onboard geotagging on embedded systems. The solution to this issue is to have a system architecture that reduces the processing redundancy, takes advantage of hardware acceleration, places telemetry into the video stream, and offers flexible operator control with standard interfaces. The thesis will attempt to solve these problems by coming up with a design and testing such a system on an embedded UAV platform.

1.4 Research Objectives

The main aim of the thesis is to improve the UAV video streaming subsystem through creating a unified and effective solution that will allow real-time video streaming and onboard recording in parallel by utilizing the use of GStreamer multimedia system. The system will address the weaknesses of currently used streamer modules by combining them in a single processing pipeline, therefore, eliminating the need to repeat computation and enhancing efficiency of the whole system on embedded UAV systems.

One of the aims is to stay on low-latency real-time video transmission between UAV and Ground Control Station. The system is also developed in such a way that it can provide ongoing live video streaming on the mission execution, despite the onboard recording being on or off dynamically. The goal is used directly in real-time decision-making in applications like surveillance, disaster response, and defence, where the timely visual feedback is important.

The other significant aim of this work is the addition of the geotagging informa-

tion based on the GPS location to the video stream. The middleware abstraction layer provides telemetry data which is then periodically obtained and inserted into the video pipeline at a predetermined update rate of about 200 milliseconds. This information can be placed directly into the video stream as visual overlays to provide synchronized spatial and temporal information of both live streaming and recorded video without the necessity of post-processing synchronization.

Resource awareness and efficiency is another key goal of this thesis. It is developed in such a way that the system can work under the computational, power, and thermal limitations of the NVIDIA Jetson Nano platform. This task is covered with the special attention to pipeline design, reuse of processing stages, and dependence on accelerated-hardware video encoding. To make UAVs practical, stability in operation when subjected to persistent streaming workloads and recording workloads is a requirement.

Lastly, the thesis will also be used to test the proposed system with extensive performance testing and evaluation. This involves testing functional correctness during the live streaming and recording, and testing performance of the systems based on the suitable evaluation measures as latency, frame continuity, resources usage and stability under diverse operating conditions. The effectiveness of the suggested approach is evidenced through the systematic experimentation and analysis in comparison to currently applicable solutions.

The combination of these goals helps to design, implement, and evaluate the proposed UAV video streaming and recording system so as to ensure that it meets the practical operational needs as well as the constraints of the embedded platform.

1.5 Thesis Outline

This thesis consists of seven chapters that discuss a particular subject of the design, implementation, and evaluation of a system to stream and record videos simultaneously with geotagging of autonomous UAVs. In chapter 1, the research topic is presented and the background of the research, motivation, problem statement, and objectives are stated on which the work is guided. It creates the necessity of an integrated and resource-efficient system of video transmission on the UAV which could provide a real-time video stream, video recordings onboard, and video integration with telemetry. Chapter 2 introduces the basic concepts that are applicable in this study. It talks about UAV architecture, onboard video processing, idea of real-time streaming, video recording format, telemetry, and geotagging systems, and embedded platforms of GPUs. This chapter gives the theoretical and technical context of this design in subsequent chapters. The state of the art of UAV video streaming systems, multimedia pipelines, hardware-accelerated encoding, and aerial geotagging solutions is also reviewed in Chapter 3. The current methods are examined and the

shortcomings of these are discussed which results into the research gaps covered in this thesis. The chapter 4 explains the system concept and the general architecture of the proposed system. It describes the UAV software architecture, describes the meaning of simultaneous streaming and recording, the GStreamer pipeline layout and the methodology applied in integrating telemetry. There is also a discussion on the design constraints and trade-offs. Chapter 5 dwells on how the proposed system should be implemented. It details the development setting, pipeline implementation of multimedia pipelines, network streaming with RTP and WebRTC, onboard recording, telemetry acquisition and overlay combination and system robustness systems. Chapter 6 gives the experimentation findings and the analysis of the adopted system. The latency, frame continuity, resource usage, and telemetry synchronization are some of the performance measures that are studied, and the system is found to compare with the current methods. Limitations that are observed are also discussed in the chapter. Chapter 7 is the last part where the thesis is concluded summarizing the main findings and contributions. It also gives possible areas of further work such as extensions of metadata manipulation, communication technologies and scalability of the system.

2 Background Research

The chapter introduces the preliminary ideas and technical infrastructure needed to comprehend the design and application of the proposed system of simultaneous video streaming and video recording in autonomous UAVs with geotagging. Although Chapter 1 introduced the motivation and objectives of the research, this chapter offers the theoretical and architectural background in which the proposed solution is developed. The chapter starts by giving a summary of UAV system architecture, which identifies functions of flight controllers, companion computers and onboard sensors. It then presents the concept of video processing in UAV and real-time video streaming, and the most used recording formats. The chapter also explains the basis of telemetry and geotagging and embedded GDP onboard processor platforms. Combined, these themes provide the foundation of the system design and implementation decisions that follow in the following chapters.

2.1 Unmanned Aerial Vehicle Architectures

Architectures Unmanned Aerial Vehicles are commonly designed to consist of several closely coupled subsystems that together permit autonomous/semi autonomous operation. On a higher level, a UAV system may be broken down into the ground segment and the airborne segment that are connected together by use of wireless connections as depicted in Figure 2.1. The airborne segment comprises of flight platform, onboard computing units, sensors, and communication interfaces, and the ground segment comprises of Ground Control Station and operator interfaces [47].

The flight controller is the heart of the airborne part and is the controller in charge of flight stabilization, flight navigation and flight control of the aircraft and so on. The flight controller works with information given by inertial sensors, that is, accelerometers and gyroscopes, and positioning sensors, i.e., GPS receivers. On this information, it implements control algorithms in ensuring stable flight and in following premeditated paths. Flight controllers are typically optimized to act in a deterministic manner and not focus on high computational throughput due to strict real time needs.

Besides the flight controller, companion computer is becoming a common component of modern UAVs to perform tasks of high-level and computational complexity. The companion computer works in combination with the flight controller and handles the duties of video processing, external system communications, mission planning,

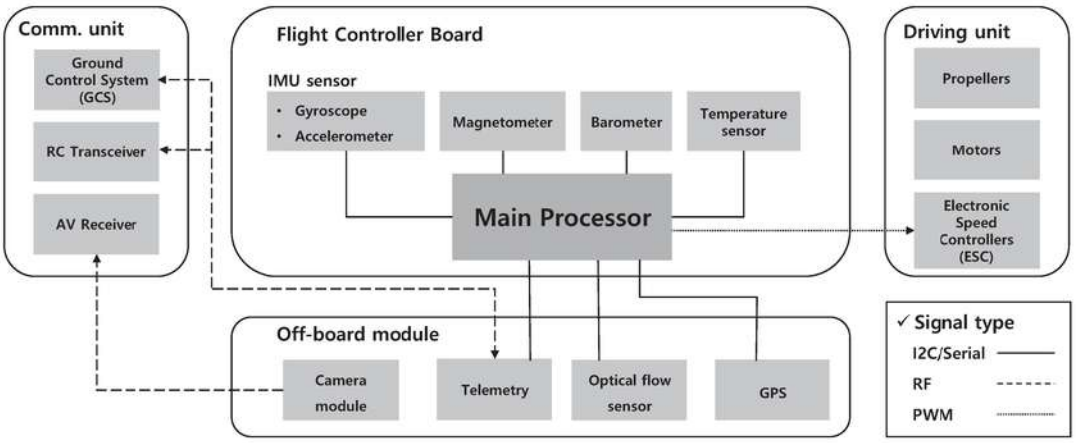


Figure 2.1: Block diagram of various modules in drone and the signal flow [24]

and data recording. The companion computer, in contrast to the flight controller, is usually a full operating system, like Linux, and can be used to support complicated software stacks and applications.

The architectural principle that ensures the flight controller and the companion computer are separated is a major one in the operation of contemporary UAV systems. Giving the companion computer some of the non-critical and high-level jobs to do will allow the flight controller to stay concentrated on time-critical control loops. This division also facilitates increased flexibility in the design of a system since companion computer software is able to be changed or expanded without altering flight-critical hardware.

Another necessary element of UAV architectures is onboard sensors. Besides navigation sensors, UAVs can also have cameras, range sensors and environmental sensors in order to sustain mission-specific tasks. More specifically, cameras produce huge amounts of data and it necessitates its own processing pipelines to support real-time streaming, recording, and analysis. Onboard video processing and less reliance on off-board computation is made possible by the introduction of camera sensors to the companion computer.

The communication between the airborne and earth segments is usually performed with the assistance of wireless networks, e.g. Wi-Fi, radio modems, cellular networks. Telemetry data, control commands and video streams are transmitted using these links. The unreliability and changeability of wireless communication means that UAV architectures need to be designed to survive latency, packet loss and changes in bandwidth particularly in sending video streams with high data rates.

At the system level, UAV architectures are shifting to the trend of modular and layered system architectures. The technique encourages the separation of concerns,

eases the process of adding new components, and enhances the problem-resistance of the system. Middleware layers are frequently added to the hardware interfaces to give a set of standard channels between the subsystems. These architectural designs are especially applicable to the combination of complicated features, such as real-time video streaming, onboard recording and telemetry synchronization.

2.2 Video Processing in UAV Systems

A basic feature of a modern UAVs is video processing, which aids in providing real time situational awareness as well as post mission processing. Video images captured aerosolically are highly graphical and are needed in usage in activities like surveillance, inspection, disaster response, and surveillance. Consequently, UAV platforms should be in a position to capture, process, transmit and store video data effectively within a rigid set of requirements concerning latency, bandwidth and resources onboard.

The UAVs video processing normally starts with the video capture of the onboard cameras which can record raw or compressed video formats. Camera output format has a direct impact on the following processing needs, such as the decoding, preprocessing, and encoding needs. Preprocessing steps, including format conversion, color space transformation and scaling, are frequently used to provide compatibility with other downstream components, as well as optimize the video stream to other processing. Video encoding is a very essential tool in minimizing the size of data to be transmitted over the air and onboard storage. The computing power of encoders can be such that numerous UAV systems are based on hardware-accelerated encoders embedded in embedded GPUs or multimedia engines. Hardware acceleration Hardware acceleration is a real-time capable encoding that consumes low CPU load and low power consumption, which may be useful in embedded UAV deployments.

The video processing pipelines of UAVs often have overlay and annotation techniques so that contextual information (e.g. timestamps, geographic position, or altitude) may be directly included in the video stream. Incorporation of overlays into the board ensures that both live streams as well as recorded videos are autonomous and can be analyzed instantly. Pipeline designs should be able to split streams efficiently to eliminate unnecessary repetition of tasks where two or more streams are needed, e.g. live streaming and onboard recording. Video processing design is also affected by the limitations of wireless communication. The video streams of the UAV are sent over links that have a variable bandwidth and latency and pipelines must be designed with low latency and strong efficient results. The pipeline design and resource management efficiency are thus necessary to guarantee a dependable functioning and other in-board functions like the flight control and the telemetry management.

To conclude, video processing of UAV systems can be described as a chain of operation, which are interconnected, such as acquisition, preprocessing, encoding, annotation, transmission, and storage. Such operations have to be well synchronized in order to achieve real-time performance needs and still perform within the limits of embedded platforms.

2.3 Real-Time Video Streaming Concepts

Video streaming in real time is one of the most important demands of numerous UAV uses because it allows the operator not only view the surrounding but also make decisions during a mission. The real-time streaming, in contrast to the offline video delivery, focuses on the importance of the timely video transmission of video frames with the minimum delay between capturing and display. This low-latency is vital in the UAV systems where applications like surveillance, disaster management, and remote inspection are highly needed because any delay in information may limit operational efficiency.

End-to-end latency: this is one of the major features of the real-time video streaming, and it is the amount of time that has elapsed between the video capture at the UAV and the display of the video at the Ground Control Station. There are several factors that affect end-to-end latency and among them is video encoding delay, buffering strategies, and network transmission time as well as decoding at the receiver. In order to minimize latency, it is common to make trade-offs, including to decrease buffering and accept the occasional loss of a packet, particularly in a wireless communication setting.

Besides the latency, the throughput and jitter are also significant in streaming performance. Throughput is the amount of video data which can be sent across a network within a specific period which directly influences the possible video resolution and frame rate. Jitter is a term used to describe the changes in the arrival times of the packets, and they can cause a disrupted playback when improperly handled. Therefore real-time streaming systems need to strike a balance between the consumption of bandwidth and buffering systems that accommodate the variability of the network without incurring too much delay. Live video streaming of UAVs is often based on time-sensitive media transport protocols, based on packet switching. These protocols are designed to favour continuous flow of data as opposed to delivery guarantees so that the system can drop old packets instead of halting the playback. This is especially appropriate in the case of live video where the latest frames are more useful than older ones.

Adaptability to network conditions is another aspect that is important in real-time streaming. UAV wireless connections might undergo variations in signal quality, interference or bandwidth capacity. Streaming mechanisms may vary the encoding

parameters, bit rates, or frame rates to sustain a constant stream in a varying scenario. Such flexibility enhances a high level of robustness, but should be applied carefully not to cause high computational overhead in an embedded platform.

2.4 Video Recording and Container Formats

The video recording is a crucial functionality of UAV systems, which assists in post-mission analysis, documentation, training, and supporting preserving evidence. Video recording is more focused on completeness, consistency, and suitability with post-processing software compared to live streaming which focuses on low latency. Recording video can enable the operators and analysts to analyze the mission execution step-by-step, evaluate the performance of the system and get the information after the flight.

Recording video on a UAV is frequently applied to relate mission results and visual evidence to the telemetry data and reconstruct the UAV flight path. This is especially relevant in the field of using it in infrastructure verification, disaster management, and policing, where the footage taken can be used as a technical report, or even a court case. Consequently, video container format option is a major factor that influences the effectiveness and dependability of captured information. A video container format specifies the way the encoded video streams, audio streams, subtitles, and metadata are coded into one file. The typical container formats found in UAV and multimedia systems are MP4, MKV, AVI and MOV. All formats vary in the types of codecs to which they support, the way that they manage metadata, their robustness, and their compatibility with playback and analysis software.

MP4 container format is popular because it is compatible with many operating systems, media players and analysis software. MP4 is a good encoder of compressed video transmissions, including H.264, and has relatively low overhead on long-duration recordings. It is also well-supported, which is why it is applicable to the UAV applications where files captured by it should be transferred, viewed, and analyzed in an easily accessible way without using any specific software. Matroska (MKV) is also another container format that is more likely to be found on multimedia systems and in which the multi- streams and rich metadata are easier to manipulate. MKV is commonly applicable to research and experimental systems where it is desired to be extended and tolerate interruptions. Nevertheless, it is not as common as MP4 to be compatible with conventional consumer tools which can restrict its viability in working operational UAV processes.

Container formats can also include subtitles and metadata tracks besides video data storage. Whereas in some systems telemetry is included as distinct metadata streams, in other systems the contextual information is included within the video frames as visual overlay. Embedding overlays also makes it easier to play and inter-

pret since the recorded video is self-sufficient, and there is no dependency on any external telemetry files, or synchronization systems.

2.5 Geotagging and Telemetry Data

Geotagging and telemetry data are considered to be important spatial and temporal context of UAV video systems. Onboard navigation sensors produce information which can be used in telemetry like geographic position, altitude, orientation and time, and so on. This information, when combined with video data, permits the correct interpretation of aerial video when it comes to real-time monitoring as well as the post-mission analysis.

Geotagging usually consists of associate video frames with GPS derived position and time data. This association would be fundamental in use in surveillance, inspection, and disaster management, where information on the location and time of visual observations is no less important than the visual data. Video frame rates are usually not equal to the telemetry update rates and it is important to carefully balance between the two to ensure that contextual information stays constant.

The integration of telemetry and video use may occur in various ways. One of them is to store telemetry apart and correlate it to the video later in post-processing, thereby adding complexity and needing more tools. Another strategy includes the implantation of visual overlays or subtitles on top of the video stream with telemetry. This approach generates video streams and video recordings which are in themselves self-interpolable, without other external information sources.

Real-time functions: Telemetry and live video allow a better situational awareness in the Ground Control Station at the point of operation. In the case of recorded video, embedded geotagging facilitates mission review, GIS visualization and documentation of evidence. The ability to effectively integrate telemetry is thus one of the design factors of UAV video systems and forms the basis of the approach taken in this thesis.

2.6 Embedded GPU Platforms(NVIDIA Jetson Nano)

NVIDIA Jetson Nano is a small embedded graphics system platform that is created to facilitate edge computing-style applications that demand efficient processing of multimedia and sensor information. It features a general-purpose CPU, a CUDA-capable graphics card, and hard multimedia acceleration units on a single system-on-chip (SoC) and provides an ideal application to UAV companion computer applications. The Jetson Nano features a central processing unit based on

the ARM Cortex-A57 quad-core processor, which is used to execute the operating system, applications logic and system integration. An inbuilt NVIDIA Maxwell (128 CUDA cores) graphics card complements the CPU to give it the ability to process data in parallel (especially video processing and other data-intensive tasks). This heterogeneous architecture enables the effective distribution of computational tasks between the CPU and the GPU as well as to enhance the overall system performance and also with low power consumption.

The major characteristic of Jetson Nano architecture is that it supports hardware-accelerated multimedia processing. The platform has special video decoding and encoding units that can support the popular video codecs including H.264. These hardware blocks facilitate the real-time encoding and decoding of videos at the lowest possible use of CPU which is critical in ensuring low latency and minimized power consumption in UAV systems. With video compression being delegated to specific hardware, the Jetson Nano is capable of supporting video streaming and recording in real-time along with other onboard operations. The memory structure of Jetson Nano is built in such a way that it supports the effective transfer of data among the processing units. The ability of the CPU, the multimedia engines to share memory and access data also eliminates data copying that does not enhance system performance and zero-copy pipelines, which are of great benefit in real-time video processing. The given architectural feature facilitates the building of efficient multimedia pipelines which reduce latency and overhead consumption of resources.

Software wise, the Jetson Nano has a Linux-based operating system and uses the NVIDIA embedded software platform, including drivers and multimedia system which opens up hardware acceleration potential to applications developers. High-level multimedia frameworks can be used to build complex video processing pipelines (without low-level hardware programming) through this software support.

In the framework of the UAV systems, Jetson Nano offers a fairly good balance between power efficiency, computational capability and physical footprint. Its architecture facilitates parallel implementation of real time video streaming, onboard recording and telemetry processing with the limitations of embedded UAV platforms. These features give the Jetson Nano a good base on which the unified video processing system outlined in this thesis can be implemented.

3 State of the Art

3.1 UAV Video Streaming Systems

A basic aspect of the modern Unmanned Aerial Vehicle (UAV) systems is real-time video streaming which allows the situational awareness of the missions in real-time [43]. Such applications as surveillance, infrastructure inspection, disaster response, and search-and-rescue are also prominent examples of video streaming using UAVs, where timely visual feedback is necessary to make decisions. Because of the unique limitations of UAV platforms, such as limited computational power on board, power supply, and wireless bandwidth, video streaming systems should be developed to reach low latency, high reliability and resource efficient operation [32].

The early studies in video streaming by UAVs were mainly aimed at the minimization of the end to end transmission latency and the maintenance of reliable communication over wireless networks. As more and more companies adopt digital video compression standards and embedded processing platforms, current research has shown that it is possible to successfully implement low-latency streaming by simply designing their pipeline and optimizing video processing phases [47]. Both of these works assert that latency is not based upon the communication channel only but by the cumulative delay added in the video capture, encoding, packetization, transmission and decoding.

The modern trends in wireless communication systems have also influenced the architecture of UAV video streaming systems. Fifth-generation cellular networks with high-bandwidth and low-latency networks have been investigated to allow real-time video transmission in addition to control signaling [19]. These strategies demonstrate that the combination of video and control data streams into one communication system can advance responsiveness and stability of the system. Nonetheless, it might be restricted by the requirement to have high-level network infrastructure to be relevant in remote or emergency situations when connectivity becomes unreliable.

Latency has now become one of the most important performance measures of live video streaming systems, and the UAV-based ones are no exception. The studies on the low-latency media streaming architecture point out that to ensure the near real-time performance, the optimization of the entire multimedia pipeline is necessary [4]. These are reducing buffering delays, support of the right transport protocols, and optimization of encoder settings. These results can be immediately applied to

the UAV video streaming where even minor time delays can have gross effects on both mission performance and occupational reaction-time.

Thorough survey research of aerial video streaming systems gives additional information on existing capabilities and limitations. The current solutions have been proven to be very effective in terms of video quality and reduction in latency, but issues of robustness, scaling, and adaptation to changing network characteristics remain a problem [50]. The results of these surveys also point at the fact that the majority of UAV video streaming studies consider the performance of live transmission, whereas onboard data management and long-term storage are not given comparatively as much attention.

This relevance of stable video streaming in real time is further indicated in the application specific deployment of UAV. Live aerial video streaming has demonstrated to improve the accuracy of assessment and efficiency in various infrastructural inspection programs by allowing them to make instant visual assessments [35]. Likewise, during disaster management situations, UAV video streaming will be used to enhance expeditious situational awareness and coordination by delivering real-time image information to emergency response teams [16]. The need to have reliable video streaming systems that can be used in dynamic and demanding conditions is highlighted through these application driven studies.

In terms of implementation, the open-source multimedia systems have contributed largely towards facilitating the flexible and effective UAV video streaming solutions. Video capture, processing, encoding and transmission Modular systems like GStreamer can offer reusable video capture, processing, encoding and transmission components and are easily adapted to embedded real-time applications [36]. They are extensible and enable hardware acceleration making it possible to design the video pipelines in accordance with the requirements and limitations of UAV platforms.

Although the UAV video streaming has been researched extensively, there are still a number of limitations that are apparent in the present state of the art. The majority of the existing systems are basically real-time transmission systems to a Ground Control Station, with the onboard video recording being added as an additional or secondary feature. Such separation can typically lead to duplication of processing operations, inefficient utilization of resources and can cause synchronization problems between the streamed video and the recorded video [60]. Besides, the present state of art performs the processing of the contextual information, which would consist of location and also the state of the vehicle and would not be analyzed to be ingested as part of the video streaming solution. These constraints have motivated the development of single video processing architecture that can be used to work with real-time streaming as well as other features on resource-constrained embedded UAV systems.

3.2 Video Streaming Pipelines

The real-time multimedia transmission systems are based on video streaming pipelines that determine how the raw video information is captured, processed, encoded, transmitted, and decoded. Video streaming pipelines The design of video streaming pipelines is especially critical in the UAV systems because real time constraints are very strict, onboard resources are limited and the condition of the wireless network is variable [45]. Studies on video streaming pipelines have thus been directed towards the optimization of each step of the pipeline to minimize latency, preserve video quality and guarantee dependable transmission [10]. Figure 3.1 shows a generic video processing pipeline architecture where the video data moves through the source components to a series of processing stages to the sink components as a reflection of the modular architecture of pipelines that are prevalently used in multimedia systems [38].

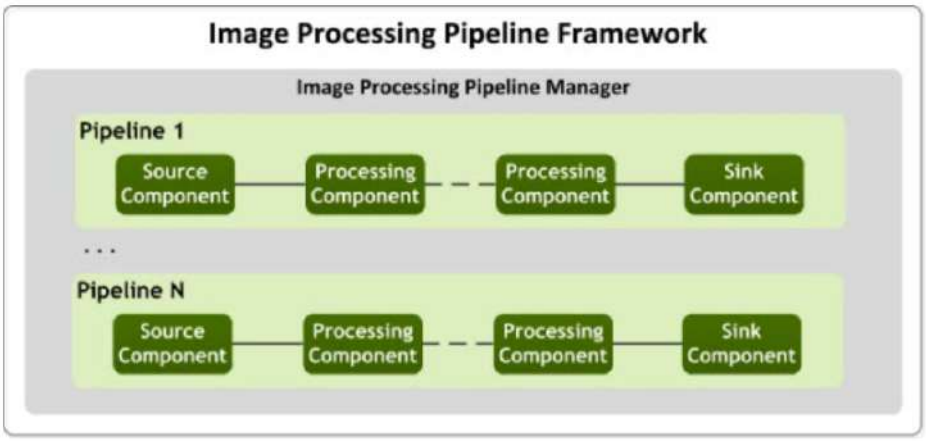


Figure 3.1: Generic Image processing pipeline framework [38]

The first designs of video streaming pipes were usually linear processing designs, with the video frame being captured, encoded, packetized and transferred in sequence [43]. Although these types of architectures are easy to implement, they can add a lot of buffering delays so they do not suit real-time UAV systems. Follow-up work placed more focus on the parallelization of pipelines and the decoupling of the stages, which made it possible to run various processing stages simultaneously and, thus, minimized end-to-end latency [47]. These schemes showed that pipeline design was as significant as encoder or network optimization in order to obtain low-latency performance.

As more wireless links of communication are used to transmit video of the UAV, adaptive and network aware streaming pipelines became a topic of research. These pipelines are dynamically controlled to change encoding parameters, bitrates or frame rates as network conditions change, in an attempt to ensure a constant video delivery [53][22]. Adaptive pipelines are beneficial in extended system robustness, however, they can lead to system complexity and need extra computational resources, which can be difficult to support on embedded UAV platforms.

The topic of low-latency video streaming has also received a lot of research in the wider sphere of multimedia, whose results are directly applicable to UAV systems. Studies on live media streaming pipelines point out that the latency depends on not just the network but also the buffering techniques, transport protocols and synchronization solutions throughout the pipeline [4]. These papers highlight the need to reduce buffering at all levels and choose light weight transportation systems and mechanisms, especially where real time feedback is needed [6].

The aerial and immersive video streaming surveys also indicate that a lot of current pipelines are focused on the performance of live transmission and ignore the other requirements of the system [50]. The onboard recording, metadata integration, and synchronization of various streams of data are usually realized as disjointed subsystems. This division may cause redundancy of processing steps and wastage of computing power, mostly in embedded systems [33][49].

The studies of application-based UAV practices have revealed that the video streaming pipeline structure has a direct impact on operation performance. A pipeline induced by too much latency or instability may render real-time video feeds of significant use in an inspection and disaster management setting [3]. These observations support the importance of strong and effective pipeline designs that could be used to work in dynamic mission environments.

In the context of this, multimedia frameworks have been invented to make video streaming pipeline construction more efficient and flexible. The most popular of these is GStreamer which has been widely used in embedded systems as well as in UAV systems thanks to its modular, plugin-based architecture. GStreamer enables the construction of video pipelines using reconfigurable elements which perform certain tasks, including capture, conversion, encoding, packetization and transmission [36]. This modularity allows the developers to develop pipelines that suit the limits and specifications of UAV platforms.

3.2.1 GStreamer as a Multimedia Framework for UAV Video Streaming

GStreamer is a widely used multimedia framework in embedded systems and UAVs that use multimedia to stream video in real-time. It has a modular and press-in architecture that lets developers assemble streaming pipelines by hooking together individual elements that perform particular functions like video capture, conversion, encoding, packetizing and transmitting packets into the network. The design allows flexibility of construction of a pipeline and is easily readable and reusable [13].

GStreamer is a video streaming application that is compatible with a variety of video codecs, video containers, and transport protocols, which makes it applicable to the various UAV streaming applications. GStreamer in embedded applications makes hardware accelerated components available to it, thus making it be able to do its processing of images with a lot of efficiency, whereby less power and less processor memory are consumed. Real-time video streaming Studies indicate that pipelines based on GStreamer are capable of being low-latency systems under the right conditions [20].

Even with these advantages, the flexibility that GStreamer has brought about poses challenges to design. It is a day frame which has several pipeline configurations and with poor selection or management of buffers can contribute to higher latency or high instability. Such challenges are best seen in cases where pipelines are stretched past the simple streaming to facilitate other functionalities like recording or metadata management.

Although multimedia systems like GStreamer offer strong systems to stream UAV video, very little of the systems capabilities are typically utilized by existing research and applications. These frameworks are mostly used in most UAV systems, primarily in live streaming, with little to no integration of other data streams or unified processing architectures. This discontinuous use causes repeated processing and resource wastage especially with embedded systems. These observations underscore the necessity of integrated framework designs of multimedia that can fully achieve the capabilities of the frameworks and overcome UAV-specific limitations. Precisely, no single solutions that synthesize real-time streaming, onboard recording, and contextual integration of data into one multimedia solution can be found. This gap is one of the main driving factors that prompted the work that is presented in this thesis.

3.3 Hardware-Accelerated Video Encoding for UAV Systems

Video encoding is also one of the key critical enablers that real-time video streaming and processing in UAV systems need hardware acceleration. The UAV platforms have very stringent limits on computational power, heat dissipation, and power consumption which restrict considerably the viability of software-based video encoding strategies. With the growing trend of using onboard perception, control, and communication activities in UAVs, the performance of hardware acceleration is currently vital to ensure the system can operate in real-time and remain stable.

Video encoding is widely based on software implementation which uses general purpose CPU, this means that the processing delay is very large and consumes a lot of power. Overhead is a potential cause of interference in mission-critical work and overall system untrustworthiness within a UAV setting. Studies of embedded computing on UAV show that multimedia processing that is CPU-intensive can often result in frame drops, longer latency, and poorer real-time reactivity [1]. As a result, the UAV systems today are beginning to use hardware-accelerated encoding to offload CPU-consuming video compression functions.

Hardware encoders are offered to offer specialized versions of popular compressed video standards such as H.264/AVC and H.265/HEVC. These encoders perform compression functions with dedicated logic, and can hence be used in real-time with much less power-usage than software encoders. Comparison of hardware video encoding and software video encoding shows that the hardware-accelerated methods are less latent and more energy-efficient and are better suited to embedded and aerial platforms [48]. H.264 is the most widely used standard in video streaming in the context of UAV video streaming since it has good trade-off between compression efficiency, latency, and wide hardware support [28].

The key role in hardware-accelerated video encoding of UAV applications is played by embedded GPU platforms. The contemporary UAV companion computers comprise of heterogeneous processing units such as CPUs, graphics cards, and video-encoding engines. Studies of ultra-fast H.264 encoding of video streams in UAVs have shown that the frame rates can be high-resolution with an achieved real-time rate and acceptable power consumption of the encoders assisted by a graphics card and hardware-based encoders [28]. These features come in handy especially when the UAV missions involve long missions that need a continuous video feed.

In addition to video streaming, hardware acceleration has found extensive use in real-time onboard UAV processing activities including fault detection, wildfire monitoring, localization and computer vision. Research into hardware-accelerated systems to detect faults and monitor the environment reveals that computing tasks

can be efficiently offloated and easier to respond to besides being more power-efficient when specialized hardware is used [57][7]. Additional examples of practical uses of specialized acceleration are event-based acceleration platforms and hardware friendly deep learning models that prove efficient as far as real-time UAV perception and tracking are concerned [8][34]. These results support the significance of hardware acceleration as a design concept of UAV onboard systems.

Hardware-accelerated components selection and integration should also be in regard to reliability and robustness. The surveys on the UAV computing platforms highlight the reliability of hardware components, fault tolerance, and thermal stability as the key parameters affecting the long-term performance of the system. Additional solutions to lowering power usage and still achieving the necessary processing accuracy in aerial imaging and video processing applications have been suggested as approximate computing and FPGA-based accelerators [37]. These tactics depict the range of hardware acceleration mechanisms studied in the UAV studies.

Although they have their benefits, hardware-accelerated encoders have some constraints. Hardware-based solutions may also have some limitations with respect to less customizable options than software-based encoders or may have fewer number of formatted modes with respect to control-based bitrates. Besides this, there may be a certain degree of reliance on driving assistance, which concerns engaged underlying hardware, limiting mobility. Consequently, hardware acceleration should be used wisely through the design of the pipeline and optimization of the system levels. Specialized accelerators, embedded GPUs, and dedicated encoding engines are highly low-latency, highly power-efficient, as well as allowing high-end onboard capabilities. The desired performance, however, is only possible when hardware encoders are closely integrated with the general video streaming pipeline. These factors are directly reflected in the system design decisions made in the following chapters of this thesis where the use of hardware-accelerated encoding is used to allow real-time streaming and onboard recording to happen simultaneously under embedded UAV conditions.

3.4 Geotagging and Metadata Integration in UAV Video Systems

Modern UAV video systems must include geotagging and metadata integration facilities that allow aerial video streams to have a spatial, temporal, and contextual association like geographic coordinates, altitude, orientation, and timestamps. This data is essential in the use of applications such as surveillance, mapping and infrastructure examination, disaster response and visualization using geographic information system (GIS). In the absence of proper metadata integration, the usefulness of

UAV video data and its interpretability are greatly diminished in the long-term.

Initial methods of geotagging in UAV video systems were mostly based on offline synchronization, in which the telemetry information and video information were collected independently during flight and later joined during post-processing. Although this approach makes real-time system design to be easy, it is vulnerable to inaccuracy in the process of synchronization and restricts the usability of real times. Studies of multimedia geotagging and spatial metadata have revealed that spatial misinterpretation can be experienced even when there is a small temporal mismatch between video frames and sensor data to the situations of dynamic or high-speed flight [31][30].

In order to overcome these constraints, some research has been conducted to directly incorporate metadata into UAV video streams. Effort to use standardized metadata formats like Key-Length-Value (KLV), to enable structured telemetry data to be added together with motion imagery is one such strategy adopted widely. By incorporating KLV metadata in UAV video streams, it has been demonstrated that the GIS-based playback and post-mission analysis have been greatly enhanced by maintaining an accurate spatial and temporal correspondence between video and flight data [14]. Those make it possible to extract machine-readable metadata without modifying the visual content of the video.

Other methods of metadata addition are the digital watermarking and data embedding schemes, in which tactical or contextual data is represented as a direct part of the video signal. Watermarking has been shown to offer real-time metadata embedding to unmanned aerial systems to allow sending tactical information along with full-motion video [44]. In some security-related applications, watermarking methods can be effective, but otherwise they can add extra processing and can complicate video compression and decoding streams.

In addition to embedding, metadata-aware video systems are also useful in retrieval, indexing and visualization. Studies on content-based retrieval systems in the UAV-like video show that, adding metadata and geographical location and time greatly improve the searchability and semantic interpretation of large video sets [39]. On the same note, the development of video content analysis and indexing of aerial surveillance points to the significance of visual features in combination with spatial metadata to aid in efficient querying and analysis [18].

Recent research has also focused on the application of geotagged video in GIS based visualization systems. Web-GIS frameworks, which are open-source, have been effectively employed to visualize geo-tagged video, and applied in road condition monitoring and infrastructure assessment to establish the utility of closely coupled video and spatial metadata [41]. These systems are based on precise and consistent geotagging of the time when the video is taken, which justifies the neces-

sity of sound metadata incorporation during UAV operation and not after processing.

Notwithstanding the progress, the state of the art demonstrates that metadata processing in UAV video systems is frequently introduced as another subsystem with respect to video streaming and video recording. This division results in discontinuous architectures and adds the possibility of synchronization errors especially when dealing with embedded systems having limited computational power. The research on geographical map marking and surveillance space also explains the difficulties of handling heterogenous metadata sources in real time [46].

Preliminary studies show that it is necessary to incorporate and align geotagging metadata and UAV video streams to facilitate proper analysis, recovery, and visualization. Although KLV metadata embedding, watermarking and metadata-aware indexing offer powerful and valuable functionality, it has been difficult to integrate it into real-time, embedded UAV video pipelines. The absence of coherent solutions that would allow the combination of real-time streaming, onboard recording, and synchronized geotagging is the reason why the integrated metadata processing strategy offered in this thesis is suggested because it is expected to work effectively despite the limitations of embedded UAV platforms.

3.5 Real-Time Streaming Approaches

Live video streaming is a component need in UAV systems that function to assist live surveillance and operator-in-the-loop decision-making. With time, various streaming strategies have been devised to be used on aerial platforms each with a different trade off in latency, robustness, scalability, and system complexity. These strategies are usually chosen depending on the mission demands, wireless network performance and the computing capacity of onboard platforms.

3.5.1 RTP/UDP-Based Streaming

One of the oldest and most widely used methods in real-time transmission of UAV video is RTP-based streaming on UDP. This model uses compressed video frames and the Real-time Transport Protocol is used to packetize them and give them sequence numbers and timestamps to enable real-time playback [15]. UDP as the underlying transport protocol enables the packets to be sent without being retransmitted; this means that the video packets that are out of date are lost instead of being held back.

The method is specifically suited in the case of UAV, where low latency is essential, including surveillance and reconnaissance operations. RTP/UDP streaming allows video to be delivered almost in real time and eliminate delays caused by

retransmission and congestion even in changing network conditions. Nonetheless, without inbuilt congestion control and error recovery procedures, visible artifacts and loss of frames may be encountered in case of packet loss [56].

In the system integration approach, RTP/UDP streaming necessitates special client program or media players that can decode RTP streams. This requirement restricts accessibility and makes deployment difficult in cases where a number of operators or heterogeneous devices is considered. Also, RTP based systems tend to only allow live transmission and onboard recordings and telemetry is still implemented as separate subsystems.

3.5.2 RTSP-Based Streaming

RTSP is an extension of RTP-based streaming that adds a layer of session control allowing to set up, teardown, and synchronize streams [42]. Systems based on RTSP enable clients to actively demand video streams and playback behavior, which can be beneficial in multi-stream or access controlled UAV locations.

RTSP has found frequent application in UAV applications, where there are predictable session management needs in inspection and monitoring applications. The RTSP has control capabilities, which make stream orchestration easier and may enhance the manageability of the system. The advantages however are at the expense of higher signaling overhead and buffering which can be harmful to end-to-end latency.

Moreover, not all current web browsers have native support of RTSP and it is necessary to use some external extension or special applications [29]. This is the limitation that limits the viability of RTSP-based solutions in web-based Ground Control Stations which are becoming more popular due to their platform independence and low implementation requirements. Consequently, RTSP has not been found to be very compatible with the current UAV systems, which value browser-native access.

3.5.3 WebRTC-Based Streaming

WebRTC has become a leading streaming way of real-time usage as the usage of web-based interfaces increases. It facilitates audio and video streaming with a low latency as well as a default web browser by integrating secure media conveyance, congestion control, and network traversal systems [25]. Such characteristics render WebRTC especially appealing to the UAV systems that should offer live video access without the need to install special client software.

WebRTC is created to work on a heterogeneous and non-reliable network, which responds to the dynamic bandwidth conditions. Research has shown that under ideal conditions webRTC can accommodate sub-second end-to-end latency which is ideal in real-time video monitoring of UAVs [54]. Besides, it is browser-native, which makes it much more accessible and scalable.

Nevertheless, the introduction of the WebRTC implementation into UAV systems adds extra complexity to the server side. RTP streams produced by the onboard video pipelines are normally to be adapted to WebRTC compatible sessions. This can require adaptation that may require implementation of intervening components like media gateways which adds complexity to the system and additional resources.

3.5.4 Media Gateway–Assisted Streaming

The media gateways are critical in connecting onboard video pipes with WebRTC customers. These gateways accept streams of RTP created by UAV onboard encoders and relay them to WebRTC receivers in addition to handling signaling, session creation, and protocol translation [61]. Media gateways allow more flexibility in the design of the system by decoupling the video processing of UAVs with the delivery to clients.

The latest generation of the lightweight gateway is especially important in the application of UAVs, which reduces the extra latency and computational load. These gateways normally do not engage in transcoding but concentrate on forwarding streams to maintain quality of video and using less resources. Conversely, advanced features include media mixing, transcoding, and large-scale distribution, which are only enabled by more feature-rich gateways at the expense of more complexity and resource consumption [5].

The gateway architecture affects the performance of the system, their scalability, embedded platform appropriateness directly. In the case of the UAV systems that have limited resources on board, the gateway assisted streaming method should be properly tooled in order to prevent the reduction of the privileges of the hardware-accelerated video processing.

3.5.5 Proprietary and Commercial Streaming Solutions

A lot of commercial UAV platforms use commercial streaming platform that closely incorporates software and hardware. Such systems tend to be designed to meet particular operation needs and may provide satisfactory performance in controlled conditions [26]. The advanced functionality embedded in proprietary solutions could

also be encryption, redundancy and vendor-specific telemetry integration.

Although these are the pros, proprietary streaming solutions are generally inflexible and not transparent. They can also be frequently limited in customization and integration with third-party components, and it is often hard to add such systems with further functionality like dynamic recording control or custom telemetry overlays. Considering research point of view, proprietary systems impede reproducibility and experimentation as well [12]. This leads to the generality of open and modular streaming methods in academic and experimental UAV systems, the extensibility and adaptability of which is paramount.

The common weakness in the studied real-time streaming solutions is that they focus on live video delivery as a single operation. Numerous are optimized to achieve low latency and stream continuity, and onboard recording and telemetry integration is undertaken as disjointed or loosely connected sub systems [40]. This division can be characterized by duplication of processing phases, high consumption of resources and coordination problems.

3.6 Research Gap

The survey of literature on UAV video systems shows that there has been a major advancement in the area of real-time video transmission, video streaming pipelines, application of hardware based encoding, geotagging and metadata incorporation. However, the analysis also demonstrates that there are still some fundamental research gaps that restrict the generalizability of available solutions to embedded, autonomous UAV platforms that need to be operated in real-time and reliable data analysis of the post-mission period.

3.6.1 Separation of Streaming and Recording

The majority of current UAV video systems are concentrated on the situational awareness, which is provided by real-time video streaming and onboard recording is handled as an independent or secondary feature [47]. This isolation tends to cause redundancy of processing steps and poor utilization of embedded hardware devices [3]. They are often configured with independent pipelines and both streaming and recording supporting which adds more complexity to the system and makes the systems less efficient in general.

Studies on video streaming pipelines highlight the importance of having a low latency scheme where the processing stages are closely integrated with strict optimization [19]. A lot of systems that are in existence however cannot maintain these optimizations when other features like recording are added. This means that real

time streaming allowing onboard recording cannot be properly served in a single, efficient pipeline.

3.6.2 Limited Integration of Hardware Acceleration

Video processing with hardware acceleration has been generally accepted as a key to real-time processing of UAV video, which is much lower in latency and power consumption. Embedded platforms that include specific processing units of encoding and GPUs have allowed complex processing operations on the board [8]. Hardware acceleration is largely implemented as a single optimization of the video pipeline (as opposed to being a unified part of the entire pipeline).

Lack of integration between hardware encoders and pipeline architecture may result in more data moves that are unnecessary, delays in synchronization and underutilization of available hardware capabilities [37]. This restricts the possible advantages of hardware acceleration, and demonstrates that system-level architectures need to be more effectively designed to include hardware acceleration.

3.6.3 Inadequate Metadata and Geotagging Integration

The meaningful interpretation and analysis of the UAV video data are entirely dependent on geotagging and metadata integration. Current methods tend to be based on offline synchronization, visual overlay or independent metadata channels. Although these techniques offer spatial and time references, they present a synchronization problem, and restrict real-time use especially on embedded systems [30][46].

Most UAV video systems focus on metadata exploitation during post mission analysis and do not address metadata processing in real-time streaming and recording processes completely. This division further risks the risk of a temporal mismatch between video and telemetry data as well as decreases the overall trustworthiness of the geotagged video outputs.

3.6.4 Lack of Integrated UAV Video Solutions

The integrated approach of streaming, pipeline design, hardware acceleration, and metadata manipulation demonstrates the absence of unified UAV video systems that cover all these demands at the same time. The solutions are usually focused on optimizing each component separately and not looking at their interaction to improve the performance and resource usage of the system. Embedded-friendly UAV video architectures are clearly required that can enable real-time streaming, onboard recording and synchronized geotagging in one single stream.

4 Concept and Methodology

This chapter introduces the idea and design of the suggested system of video streaming and recording with geotagging in autonomous UAVs working simultaneously. Expanding on the drawbacks observed in the state of the art, the given approach assumes the use of a single software architecture integrating real-time video processing, telemetry processing, and network streaming on the base of a single embedded system.

The autonomous UAV systems in the modern world are generally based on a layered software architecture that includes flight control layer, a middleware or communication layer, and high-level application layer. Real-time stabilization, navigation, and low-level sensor fusion is carried out in the flight control layer, which is typically thought to be a flight control dedicated to such purposes. High level application layer It is an application layer that operates on a companion computer and performs tasks that are computationally intensive like video processing, mission logic and communication with external systems. This separation provides safety in the flight, and flexibility in relation to application development.

The offered system is brought to the companion computer and it adds to the traditional video streaming (STR) element the onboard recording and geotagging capabilities. Rather than entering the process of streaming, recording, and telemetry visualization as individual and disjointed subsystems, the approach implemented in this thesis considers these functions as one and the same multimedia processing pipeline. The rationale behind this design decision is to minimize redundant processing, enhance the synchronisation of data streams and to achieve effective utilisation of the limited embedded resources.

In the general architecture, the video information is obtained via an onboard camera and the multimedia subsystem processes it, whereas the telemetry information like spatial position and attitude of the vehicle is delivered by the navigation and sensor subsystems via a middleware interface. The Ground Control Station (GCS) is a remote client, which receives live video images and gives control commands to the UAV system. The communication between the UAV and the GCS is achieved by regular network communication systems which allow the platform-neutral access to real time video data. The architecture of the UAV software at high level provided in Figure 4.1 depicts how the UAV hardware and sensors interact with the onboard software system, which in turn interacts with the ground control station where live video monitoring and controlling are carried out.

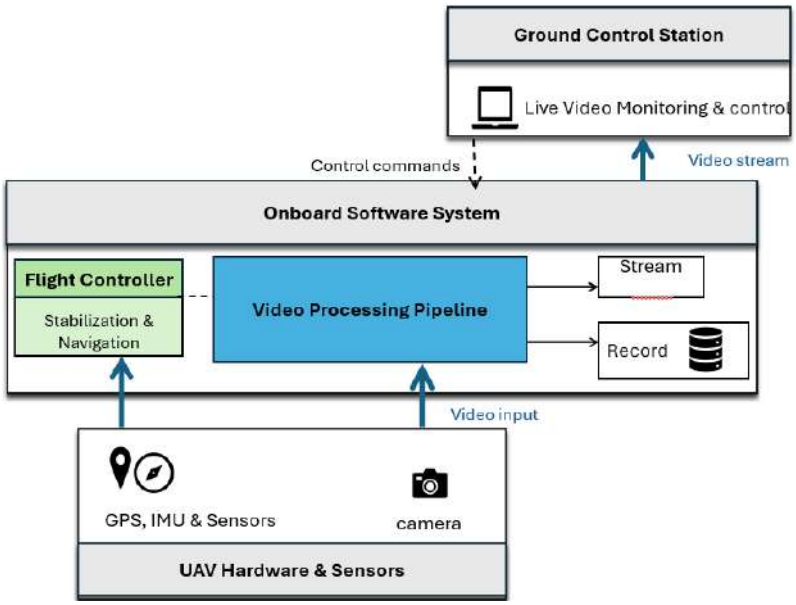


Figure 4.1: UAV Software Architecture diagram

Methodologically, the architecture focuses on modularity, extensibility and efficiency of resources. All the functional blocks such as video acquisition, processing, streaming, recording and integration of telemetry are well divided at the architectural level but closely integrated with the unified pipeline architecture. This contributes to the system being scaled to other different UAV platforms, cameras, or communication connections with only a few modifications to the fundamental framework.

Overall, the general UAV software architecture offers the organizational basis of the proposed system. The approach places simultaneous streaming, recording and geotagging as a single companion-computer-based architecture, thus bridging the research gaps found in the literature and suggests the setting up of the detailed design concepts of the following sections.

4.1 Concept of Simultaneous Streaming and Recording

This thesis has made a major contribution to understanding of methodology in that it designed a system that allows real-time video streaming and onboard video recording concurrently using only one camera source in an autonomous UAV. In most traditional UAV systems, live video streaming and onboard recording are provided as disjointed, and usually independent processing lines or redundant encoding

steps. Although they might meet functional needs, these designs cause more computational load, power consumption and possible discrepancies between streamed and recorded video data.

The idea presented in this paper is founded on one integrated video processing pipeline that considers streaming and recording to be the outputs of one processing pipeline instead of two independent subsystems. The only input to this pipeline is video frames that have been taken onboard by the onboard camera. This construction will make sure that the further processing operations will be based on the consistent and synchronized video stream, which will lie at the foundation of the reliable real-time transmission and correct data storage.

After being acquired, the camera video is forwarded through a video processing pipeline in which necessary preprocessing functions are executed. Such functions involve format translation, image sharpening, and addition of image overlay (timestamps and geotags). These operations should be carried out at an early stage so that the live stream and the recorded video can consist of the same visual information, which should ensure the same consistency of real-time monitoring and post-mission analysis.

The video stream is preprocessed and an encoded video stream is created with the help of a hardware-accelerated video encoder. A single encoding stage is a methodological decision that is used in order to reduce unnecessary calculation. Encoding remains one of the most high resource consuming processes in a video pipeline especially in embedded systems. The system is able to achieve this through the encoding of the video once and sharing of the resulting compressed stream ensuring that real time can be achieved with a major reduction in CPU and GPU usage. Figure 4.2 is the basic concept diagram for simultaneous streaming and recording.

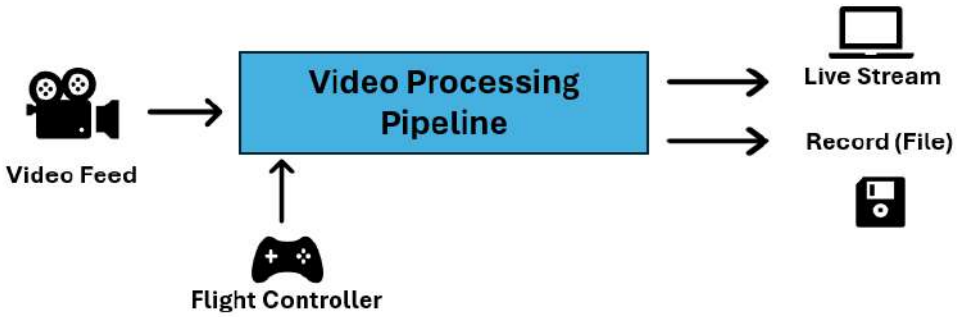


Figure 4.2: Basic Concept of Streaming and Recording

After the encoding phase the pipeline is divided logically into two parallel output streams allowing it to run in parallel. In the former, the path is devoted to the streaming of live videos, in which the coded video is sent to the Ground Control Station (GCS) via the low-latency network connection. This streaming route is created to be timely rather than flawless so the operator is able to be provided with the current visual data even when the network conditions are variable. Situational awareness to enable autonomous and semi-autonomous missions of UAVs is supported by the live stream.

The second route is allocated to onboard video recording in which the same encoded video stream is stored in persistent onboard storage. The video is saved in a container format of Matroska (MKV) which is highly suitable when using long recordings and is very resistant to unforeseen interruptions like power outages or system reboots. By saving the video to the local memory, it is guaranteed that high-quality videos will be maintained regardless of what the state of the communication connection is in the air.

There is another significant point in this concept; streaming and recording work simultaneously and independently after the splitting of the pipeline. Live streaming is not reliant on the recording process and vice versa. This separation helps to make sure that a transient problem in one output path such as storage capacity or network problems does not spread to the other path. Meanwhile, the two streams are in sync as they are based on the same encrypted video stream.

It also promotes dynamic control of recording process based on the concept. Although live video streaming usually can be performed continuously during the mission, onboard recording can only be necessary at certain mission points, e.g., during inspection or emergent events. The approach thus permits recording to be turned on or off at run time without breaking the live video pipe or any re-initiating of the pipeline. This is especially significant in UAV operations where the storage space is minimal and the needs of the mission are subject to dynamism.

Along with the onboard storage, the video files of the recorded MKV could be transferred to the Ground Control Station either after the mission or during the operation based on the bandwidth and the operational requirements. This has made it possible to analyze the post-mission in detail, record, and archive the visual information. Since the video recorded already has all the visual overlays that are of interest, it can be viewed without having to access another set of telemetry logs or synchronization tools.

The proposed concept in its methodological perspective tackles some of the key issues that were identified in the state of the art. The design of the system eliminates repetition of pipelines and repetition of encoding steps and thereby the system enhances efficiency and scalability of the resource on embedded UAV platforms. It

ensures that there are visual similarities between streaming and recording video by inserting overlays before encoding. Finally, seeking to integrate recording with streaming into the same streaming pipeline makes the concept of streaming and recording easier to design, and more robust and flexible.

4.2 GStreamer Pipeline Design

The coherent idea of streaming and recording that was outlined in Section 4.2 is fulfilled with the help of the architecture of the modular pipeline with the use of GStreamer multimedia framework. GStreamer is chosen because its processing paradigm is a graph, multimedia processes can be presented in the form of interconnected components with distinct roles. The design paradigm promotes low latency usage, modularity and extensibility hence fits embedded UAV systems well.

At the architectural level, the GStreamer pipeline has a directed processing graph that takes Raw Video data of onboard camera and converts it to various synchronized outputs. The pipeline starts with a video source element which communicates with the camera hardware and delivers raw video frames which are uncompressed. These frames are the Raw Video block that is present in the pipeline diagram and is the typical input of all the processing stages that follow it. The suggested GStreamer pipeline, as shown in Figure 4.3, includes the splitting of the streaming with tees to allow live streaming and onboard recording simultaneously.

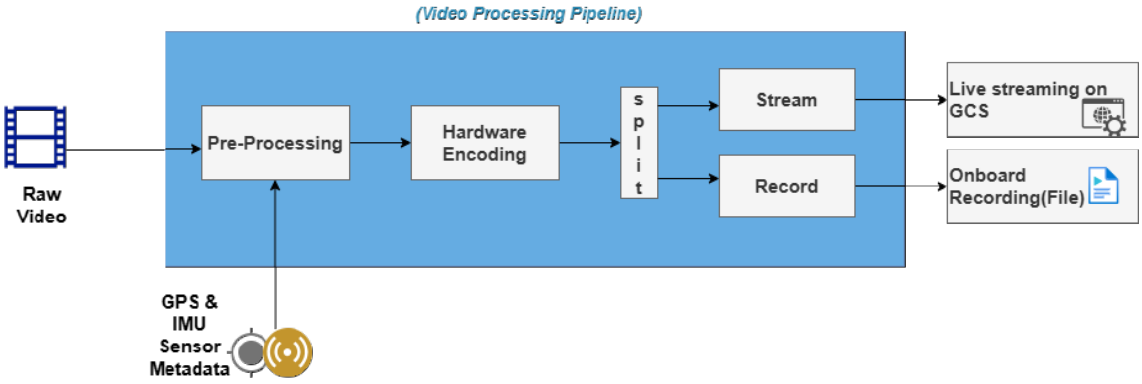


Figure 4.3: GStreamer pipeline design concept diagram

Raw video frames are fed through a Pre-processing and Geotagging step where necessary video format conversion and prep processes are performed in order to make them compatible with the subsequent components. This is also the logical point of integration of visual overlays such as timestamps and geospatial content that is directly rendered on video frames before compression. Conducting preprocessing and

geotagging in this step is important to make sure that the contextual information in both live and recorded outputs is the same.

The video stream is preprocessed and further sent to a Hardware-accelerated Encoding stage where the video is compressed by the H.264 standard. The single hardware-accelerated encoder is a conscious design decision that would reduce the amount of computational overhead and minimize the amount of power used and provide a uniform video representation to all outputs.

The pipeline then uses a GStreamer tee (splitting) element, using which the stream of the encoded video can be logically duplicated into a single stream into a number of parallel streams. The tee aspect is the major component of the partitioning of the pipeline, and the downstream branches may run autonomously without further encoding or preprocessing required. This component is directly related to tee (separating) block shown in the pipeline diagram. Below the tee element, there are two major branches. The first branch is the Packetization and Transportation phase during which the encoded video stream is ready to be transmitted to the network by real-time transport protocols. This branch ends at the Live Stream output which provides low latency video to the Ground Control Station to be real-time monitored and controlled.

The latter branch is linked to the Multiplexing stage, during which the encoded stream of video is put into an adequate container format to be stored. This branch ends at the Save File output, which is the format in which the video is saved onboard in Matroska (MKV) format. The video tapes thus captured can then be moved to the Ground Control Station to post-mission analysis and archive. Decoupling of downstream branches is an important architectural benefit of the tee-based design. Any delays caused by network issues when using the Live Stream do not affect the Save File operation and any delays occurring when using the storage do not affect the transmission of the real video in real time. Meanwhile all the upstream processes between Raw Video acquisition and Hardware-accelerated Encoding are common to enhance the effective use of the available resources and natural timing alignment of the streamed and recorded video.

4.3 Video Streaming Protocol Selection for UAV Systems

An effective video streaming protocol is a critical design choice when designing a video system based on UAVs; these choices directly influence the latency, robustness, scalability and usability of the video system. In contrast to the traditional multimedia applications, UAV video streaming has to be run with strict real-time require-

ments and utilize wireless communication connections which tend to be erratic and bandwidth constrained. Also, more recent UAV systems are using browser-based Ground Control Stations, which add additional requirements to protocol compatibility and client-side accessibility.

System design wise, video streaming protocols specify how to transfer coded video data in the form of packets over the network, how to buffer the video at the receiver and how to render the video to be viewed. Such protocol-level choices affect how the quality and responsiveness of the video is perceived as well as affect the design of the UAV video processing pipeline in general. As a result, the choice of the protocol should be viewed as an inseparable part of the system approach and not an implementation consideration.

The UAV systems do not have a streaming protocol that can be optimized to achieve all the desired features, which are: ultra-low latency, high reliability, browser compatibility, and the efficient use of embedded resources. Consequently, recent UAV architecture designs have tended to take up a layered or hybrid style in which more than one streaming technology is integrated to cater to various functional needs in the same system. Here, the most topical streaming protocols that may be used in the UAV applications are discussed, and their appropriateness is evaluated concerning the real-time operation, constraints of embedded platform, and compatibility with browser-based Ground Control Stations.

4.3.1 Requirements for UAV Video Streaming Protocols

The UAV video streaming protocols have to meet a set of specific requirements that are quite distinct as compared to the traditional multimedia delivery infrastructure. To enable real-time situational awareness and operator-in-the-loop decision-making, first of all, low end-to-end latency is needed. Delay of several seconds, which might be tolerable in video-on-demand, would not be acceptable in UAV missions which include surveillance, navigation support or emergency response.

Second, the UAV streaming protocols should be resilient to the fluctuating wireless network environments. Communications between UAVs are prone to interference and signal attenuation as well as changing bandwidth rapidly as the platform traverses the environment. Streaming protocols should then be able to withstand loss of packets and changeable network conditions without causing excessive buffering or interrupt of the stream.

Third, embedded video pipelines compatibility is one of the major requirements. The companion computers that are used in UAVs usually utilize video encoders that are hardware-accelerated and multimedia frameworks producing real-time video streams in compressed form. The streaming protocols have to be able to interact

with these pipelines with a minimal amount of transcoding overhead, and without needless processing steps.

Lastly, newer UAV systems are moving towards simpler Ground Control Stations, based on a browser, which is easy to deploy, and more accessible. Streaming protocols with native browser support allow operators to watch live video through normal web technologies, without the use of special clients. The combination of these requirements is the strategy of selecting the protocols that are used in the proposed system.

4.3.2 RTP-Based Streaming over UDP

One of the most straightforward and latency-efficient systems of video transmission is the RTP-based streaming over UDP. This model is based on the encoding of video frames into packets with the help of the Real-time Transport Protocol and the transmission of the packets through the connectionless UDP transport. RTP also offers sequence numbering and timestamping which allows receivers are able to reassemble media streams in real time and UDP also reduces protocol overhead by not requiring retransmission and connection control.

Technically, RTP/UDP streaming is ideally applicable in internal communication low-latency within UAV systems. It can be used in conjunction with embedded multimedia pipelines and hardware accelerated encoders, so that the behaviour of packetization and buffering can be fully controlled. This renders RTP/UDP especially appealing to the transmission of the video between on-board components or between the UAV and the processor components in the back-end.

Nevertheless, RTP/UDP streaming is characterized by a lack of congestion control and is not supported properly in the modern web browsers. Client-side consumption normally needs specific media players or bespoke software and this makes it less accessible and difficult to implement. In addition to this, session management and scalability is to be managed externally. Consequently, RTP/UDP streaming can be used successfully to transport the backend, but is not a feasible application as an independent solution to browser-based Ground Control Stations.

4.3.3 WebRTC for Browser-Based Real-Time Streaming

Web Real-Time Communication (WebRTC) has become one of the main solutions to the low-latency delivery of media over the web. WebRTC provides real-time audio and video streaming through the browsing experience of common web browsers as a result of integrating secure media transport, adaptive congestion control and intrinsic support of network traversal. WebRTC has all these characteristics, which

is why it is especially appealing to UAV systems that need live video visualization and do not require any specialized client software.

The adaptive bitrate and congestion control features of the WebRTC are a major technical merit since they automatically modify transmission behavior according to real-time network feedback. This is particularly useful when operating in UAV conditions where the quality of wireless links may vary quickly as the air vehicle moves, or as a result of interference or the environment. WebRTC can be used to adjust to the conditions of the network and ensure continuity of streams, with a low number of latency spikes.

WebRTC also includes common technologies of firewall and network address translations traversal, to allow media delivery on heterogeneous network foundations to be dependable. Architecturally, WebRTC makes the client-side implementation much easier by taking advantage of the support provided by native browsers and Ground Control Stations can be implemented as a lightweight web-application.

Regardless of these benefits, WebRTC was not developed to interface with common embedded video encoders. RTP streams normally produced by UAV onboard pipelines do not include the signaling and session semantics that is needed by WebRTC. To close this gap, intermediary elements that can be able to deal with the signaling, session negotiation and protocol adaption are needed. Although this adds more complexity to the architecture, the advantages of WebRTC as far as accessibility and real-time performance is concerned, make it worth using in the proposed system.

4.3.4 HTTP Live Streaming (HLS)

The HTTP Live Streaming (HLS) is an adaptive streaming protocol which is aimed at transmitting multimedia data over regular HTTP connections in such a way that it could be reliable. HLS works based on the principle of segmenting a steady video stream into a series of small bits of media, usually a few seconds in length and supplied one after the other to the client. Accompanied with these segments, a playlist file indicates the quality levels and sequence of the segments so that the clients can dynamically choose the right representations based on the network conditions.

Technically, HLS has more robustness and compatibility than latency. Using HTTP as the transport mechanism, HLS can cross firewalls and network address translation settings with limited setting up. It is also popular among the video distribution tools of big scale, as it has great support in most web browsers, operating systems and media players.

But, the segment-based HLS nature implies a buffering delay in principle. Multiple segments are normally buffered by clients prior to playout to facilitate continuous rendering with the latency at the end of end-to-end being several seconds to tens of seconds. Although there are low-latency additions to HLS, these additions also create delays which are not acceptable when it comes to real-time UAV systems that need real-time feedback and operator communication.

The HLS is not suitable in UAV systems in terms of live video streaming of critical missions in real time. However, it may be useful in the post-mission analysis and dissemination of videos. Indicatively, captured UAV videos can be converted to HLS format, which can be easily viewed by using web browsers or mobile devices. In this case, the latency limitation of HLS takes second fiddle to its scalability and strength.

4.3.5 Dynamic Adaptive Streaming over HTTP (DASH)

Dynamic Adaptive Streaming over HTTP (DASH) is an international adaptive streaming standard of multimedia that has a lot of conceptual similarities with HLS. Similar to HLS, DASH transmits video in the form of a sequence of fragmented media files through HTTP so that clients can adjust the media quality to their current network properties. DASH is more flexible in its support of codecs and structure of segments allowing more fine-grained control of video representations.

System design wise, DASH would be very appropriate where the conditions of the network are not predictable and quality consistency is of utmost importance. Its standardized format enables it to be interoperated in various platforms and devices making it a universal preference when it comes to video-on-demand, broadcast-style applications.

In spite of these benefits, DASH also has the use of buffering and segmented delivery which inherently adds to the latency. DASH does not obtain sub-second latency needed to support real-time UAV operations even in optimized settings. Consequently, DASH cannot usually be applied in those applications where operator feedback is necessary, including live surveillance or remote piloting assistance.

DASH can be regarded in the context of UAV systems, which can be offline video inspection, archiving, or analytical mission data replay. It is adaptive and can guarantee quality playback in different client devices and network conditions. Nonetheless, as in the case of HLS, DASH does not fit into the real-time streaming route of the suggested system because of its latency nature.

4.3.6 Protocol Selection Strategy in the Proposed System

According to the analysis above, the proposed UAV video system will use a layered technique in selecting protocols; the protocols in the proposed solution will be a combination of various streaming methods to satisfy different needs of the system. RTP/UDP streaming that can be used on low level onboard and backend components to support efficient and low-latency transport of encoded video data. It works very well with hardware-accelerated video pipelines and reduces the processing cost on the embedded platform.

To deliver the WebRTC to the Ground Control Station, WebRTC is chosen as the red-letter streaming protocol because it has low latency, adaptability, and compatibility with the native browsers. This option allows displaying video in real-time and interacting with it in real time with an application of standard web tools, which is consistent with the design requirements of the system in terms of accessibility and facilitation of implementation. Segment based protocols like HLS and DASH are specifically avoided in the real time streaming path because of their latency properties, although they may still be useful in the case of non interactive access to videos.

System Component	**Protocol**	**Purpose**	**Reason**
Onboard video pipeline	RTP/UDP	Internal video transport	Very low latency and efficient integration with hardware-accelerated encoders
Backend / gateway interface	RTP/UDP	Stream forwarding	Lightweight transport with minimal processing overhead
GCS (browser)	WebRTC	Live video visualization	Low latency, adaptive to network conditions, and native browser support
Onboard recording	MP4 (H.264 in MP4 container)	Post-mission storage	High compatibility and ease of playback and analysis

Table 4.1: Protocol selection strategy in the proposed UAV video system

The allocation of streaming protocols between the system components as a result is highlighted in Table 4.1 as the selection of each protocol is based on the latency requirement, deployment constraints, and the expected usage in the proposed UAV video system. The proposed system integrates these protocols into one

architecture, which allows the protocol to use strengths of each method and address the weaknesses. This protocol selection scheme has a direct connection to the system goals of continuous real-time streaming, on-demand onboard recording, and visualized telemetry, and is within the capabilities of embedded UAV platforms.

4.4 Geotagging Integration Concept

Geotagging will be the core of the proposed system because it will serve to enrich aerial video data with context information that will enhance situational awareness and post-processing. Geotagging in the given implementation is implemented by a two-pipeline architecture where the video processing and telemetry processing are two concurrent independent pipelines. Camera acquisition, processing, hardware encoded encoding, and further division into live WebRTC streaming and local MP4 recording streams are all done by the video pipeline. Parallel to this, another telemetry pipeline receives telemetry data on UAVs via UDP, processes this data via a Python-based service and shares the data in real-time to be visualized and stored in a persistent manner. Telemetry data sources are sent to the web interface through Server-Sent Events (SSE) to display live, and at the same time stored in subtitle (SRT) and structured JSON. Video and telemetry pipeline synchronization This is accomplished by a common time reference so that the video and the recorded telemetry share identical presentation and recording with the video material correctly aligned to the telemetry data and the metadata is not stored as metadata directly in the bitstream of the video.

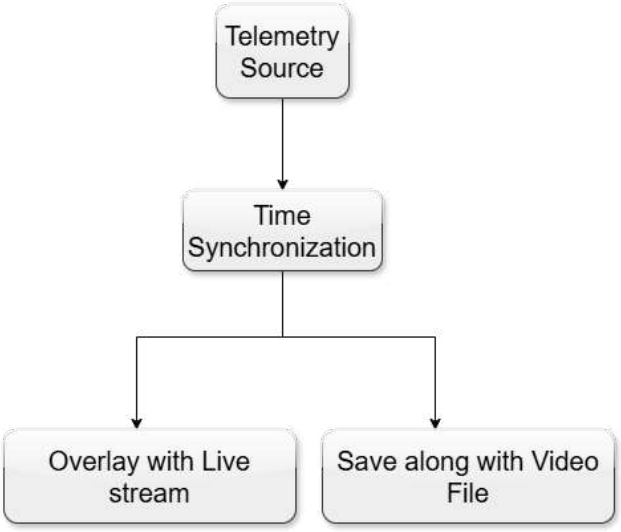


Figure 4.4: Concept of Geotagging

Various architectural approaches can be used to achieve geotagging integration

into UAV video systems with a different trade-off that includes a different level of accuracy in synchronization, complexity of the system, accessibility, and robustness. The choice of a geotagging strategy has a direct impact on the mechanism of capturing telemetry data, its storage, and visualization in regard to video content. According to the prevalent UAV video systems and multimedia processing practice, the integration methods of geotagging can be divided into three general methods.

4.4.1 External Metadata Association

The most time-honored solution to geotagging integration is external metadata association, in which the telemetry data is archived independently of the video stream. In this approach, the navigation and sensor information including GPS positions, altitude, and attitude are stored in external files or databases and the video is stored separately. During post processing, synchronization between the telemetry data and video frames is done with the help of the timestamps.

This method will leave the original video material intact and the telemetry information will be processed or analyzed separately. Nevertheless, it brings a lot of complexity to the data management and the use of the video and telemetry subsystems needs to be synchronized closely in terms of time. Any time difference or imprecision in the timestamps may cause discrepancies between the video frames and its context. Moreover, the geotagging information does not necessarily accompany the video file, so the contextual information can be lost in case the video is opened or shared without its metadata files.

4.4.2 Embedded Metadata Tracks

A more comprehensive solution is to store the telemetry information in the video container as a special metadata track. This is usually done with a standardized metadata format (i.e. the Key-Length-Value (KLV) format) where structured telemetry data can be multiplexed with the video stream. Under this approach, geotagging information follows the video and can be decoded or rendered in the video playback using compatible software.

Metadata insertion maintains the video and telemetry separation and still ensures that they are synchronized, so it is appropriate in an application where accuracy in space and time is paramount. Nevertheless, with such a system, complexity is raised, and it relies on specialized playback software with the ability to read the metadata tracks. Embedded telemetry is not a universal concept and is not supported by all media players and browser-based environments, which reduces its practical application on lightweight Ground Control Stations. Moreover, such metadata may need

bespoke visualization to display in real-time.

4.4.3 Visual Overlay-Based Geotagging

The third method is that of visual overlay-based geotagging in which the telemetry data is overlaid on the video frame as text or graphical overlay. Within this approach, the information on geotagging is involved in the visual content and cannot be separated together with the video stream. The overlays are produced in the video processing pipeline, as is usually part of an earlier stage of the processing (before encoding).

Geotagging using visual overlays has a number of benefits. It makes sure that the context is not lost no matter what the playback environment or software to be used is and it is universal and strong. This simplifies the architecture of the system because parallel streams of metadata are not required or post processing synchronization. It however compromises the fact that it can access raw video material without overlays and restrict the flexibility of extracting post-hoc data.

Regardless of such constraints, visual overlays are especially practical in real-time UAV applications, where the contextual information is more valuable than video-metadata separation. This simplicity and reliability of this method encourage the use of embedded systems and browser-based Ground Control Stations.

Geotagging Method	**Post-Mission Analysis Capability**	**Resource Utilization**
External metadata files (JSON / SRT)	High (machine-readable, easy correlation with video frames)	Low to moderate (no impact on video encoding pipeline)
Embedded metadata tracks (e.g., KLV)	Moderate (requires specialized tools for extraction and analysis)	High (additional parsing, muxing, and decoding overhead)
Visual overlay + synchronized external telemetry (proposed system)	Very high (human-readable overlays + structured telemetry logs)	Low (hardware-accelerated video pipeline remains unaffected)

Table 4.2: Comparison of geotagging approaches with respect to post-mission analysis and resource efficiency

Table 4.2 includes a comparative summary of geotagging integration methods in terms of the ability to provide a post-mission analysis and efficiency of the system resource consumption. The findings indicate that the various geotagging practices have different trade-offs in respect to the flexibility of the analysis, the overhead of the processing, and usability of the operation. External metadata association provides good support on post-mission analysis with the help of structured telemetry files, but it requires accurate timestamp synchronization between the video and telemetry streams. Embedded metadata schemes, like KLV-based integration, offer closely bound metadata at a high cost, and have the disadvantage of only being playable in a limited set of decoding settings. Conversely, the proposed system is hybrid based on the combination of real-time visual overlays and external telemetry that is synchronized. Thus, the visual overlay-based geotagging approach used in the present project offers the best balance between the analytical strength, the capability to play back the information, and the ability to use resources effectively, which is why it is well adapted to conducting real-time UAV jags and analyzing the gathered information offline.

4.5 Design Constraints and Trade-offs

A variety of technical and operational limitations has an inherent effect on the design of a system capable of delivering simultaneous video streaming and recording of autonomous UAVs with geotagging. These limitations are due to the nature of the UAV platforms, the real-time demands of video streaming, and the demands of good data capture in dynamic environments of the missions. The suggested system architecture and approach constitute a progression of trade-offs made to balance between performance, flexibility, and robustness.

Among the major limitations that have been factored in this work is the limited computational and power that can be operated on the UAV companion computer. Video processing more so encoding is a computationally intensive task. To overcome this limitation, the system takes the single hardware-accelerated encoding step which is common to all output paths. Although such design is very efficient in terms of resources and minimizing power usage, it also limits all outputs to the same encoding parameters. The trade-off is not objectionable in the light of the proposed system, as it would guarantee consistency in live and recorded video and also allow it to be in real time.

The other important limitation is associated with the real time latency requirements. Video streaming of UAV operations is not very much concerned with the reliability but with timely delivery. The system thus prefers the low-latency transmission schemes that can accommodate the occasional losses of packets in case of unfavourable network conditions. This design solution will enhance the situational awareness in the air but at the expense of assured video delivery. The effect of

such trade-off is offset by onboard recording, which stores high-quality video data regardless of the network performance.

The decision to use Matroska (MKV) as the onboard recording file type is an indication of a compromise between compatibility and robustness. MKV has the advantage of being resilient to interruptions and available to support long-duration records, so it is suitable in UAV missions. MKV files however can be processed further or converted to be able to work with some playback or analysis software. Such a trade-off is deemed to be acceptable in the light of the focus on the reliability of data capture and accessibility after the mission.

The addition of geotagging brings more designing factors. The fact that telemetry information can be embedded into the video stream as visual overlay will help to ensure that the geospatial context will be present at all times throughout the playback. This however minimizes flexibility as compared to storing metadata using separate tracks because the information embedded can never be edited or removed once the tracks have been recorded. The compromise between the operational resilience and the flexibility of post-processing that goes with burned-in overlays supports this option.

A tee-based pipeline splitting mechanism brings about a trade-off between the simplicity of the pipeline and lack of dependence on the output. Although tee is able to readily replicate the video stream that is being encoded, downstream branches are still at the mercy of the stability of upstream pipeline phases. The problem is thus that design of the pipeline must be careful so that delays or failure in one of the branches does not spread upwards. This trade-off is resolved in terms of the pipeline decoupling strategies and independent downstream processing which have been presented in the above sections.

Finally, a concern of the system design is its emphasis on modularity and extensibility rather than a highly optimized system with a specific goal or proprietary hardware or proprietary software development kit. Even if there is a potential performance benefit from a more proprietary system with hardware or a proprietary software development kit, this would negatively affect system transparency and flexibility, which is contrary to the research-based objectives of the current thesis topic.

5 Implementation

This chapter discusses the implementation of the proposed architecture on the embedded UAV companion computer on the constraints of computational power and the necessity to make the system robust and stable which exist in the real world. The previous chapter discusses the concept of the proposed system to possess the capability of video streaming and recording simultaneously with the option of geotagging on the autonomous UAV, which focused on the description of the proposed system and the methodology on the architectural level.

The implementation is based on a modular design pattern, which is almost identical to the unified pipe architecture described above. Activities such as; video acquisition, video preprocessing, video encoding, video streaming, and geotagging are broken into well-organized and logically segmented pieces in one multimedia processing pipe. This helps in efficient reuse of processing entities and synchronization of live and recorded video streams. Close attention has been given to the application of hardware acceleration, as well as, effective multimedia frameworks to low latency on embedded systems. In addition, there has also been an inclusion of runtime control, handling and tolerance mechanisms to the system so that realistic UAV missions can be conducted. Figure 5.1 shows the overal implementation steps in the thesis. The following sections will address the development context and details of implementation of each part in the system.

Following this, the chapter will initially describe the environment where the proposed system has been deployed, that is, the embedded hardware components and the software environment. It will further move to the description of the implementation of video input processing and subsequently the processing of video compression using hardware to attain real-time compression. The next step of the chapter will be the description of the branching of the video pipeline to obtain the capability of live streaming and recording, and then the description of the implementation of the network transfer and storage. The chapter will then elaborate how the telemetry information is realized in terms of its integration by means of the retrieval and rendering of its values. Lastly, the chapter will explain how error handling will be realized.

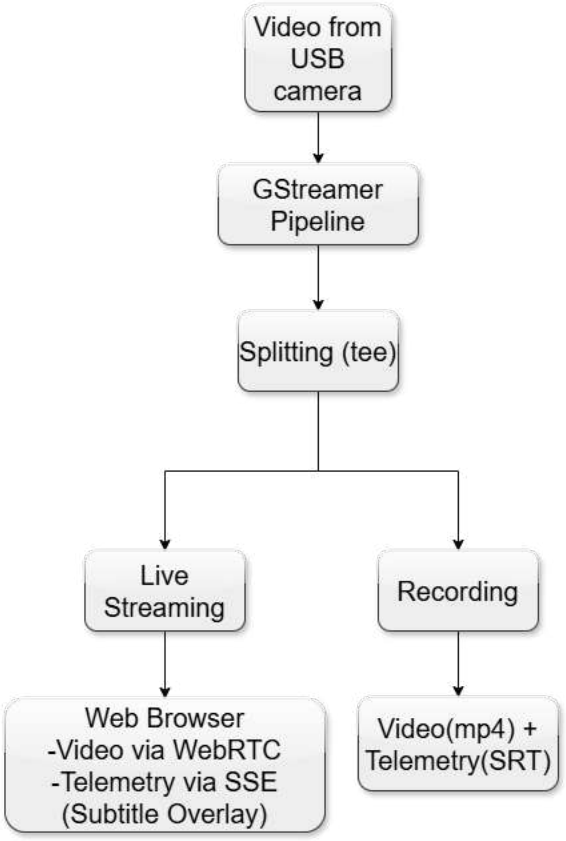


Figure 5.1: Implementation Diagram

5.1 Environment Development

The development platform is the basis on which the proposed system is developed and tested. The selection of hardware and software components is critical to the proposed system to guarantee the real-time processing capabilities while maintaining the practicality of the system as it would be on an autonomous UAV. This section discusses the hardware platform on which the system is developed and the software components.

5.1.1 Hardware Platform: NVIDIA Jetson Nano

The deployment of the proposed system is done using the NVIDIA Jetson Nano, which will be used as the companion computer of the UAV. The NVIDIA Jetson Nano is an embedded computing board with low power consumption, and it is appropriate for real-time multimedia applications. Its size, power, and processing capabilities match the requirements of the UAV.

The hardware components of the Jetson Nano include the quad-core ARM Cortex-A57 processor and the NVIDIA Maxwell graphics processing unit, which is composed

of 128 CUDA cores. The combination of these components makes it possible to run control and graphics-intensive computations concurrently. In the proposed system, the capability to run computations concurrently is important because the video processing pipelines require control and graphics-intensive computations.

The other feature of the Jetson Nano that is relevant to the proposed research is the capability to encode and decode video hardware. The hardware allows the use of special multimedia processors so that it can support real-time video compression and decompression with the H.264 and H.265 algorithms. The software video encoding feature is relevant to the proposed study as it allows the system to stream the video and record it at the same time.

On a data processing perspective Jetson Nano has been optimized with memory paths that can be used to efficiently transfer frames between camera interfaces, GPU memory, and encoder hardware. It comes in handy to avoid redundant copies of frames within the video processing pipeline, which is directly coupled to the integrated pipeline architecture that the work of this thesis has.

The Jetson Nano provides a variety of interfaces that would be appropriate for integration with UAVs, including camera ports, Ethernet networking, and general-purpose I/O. Furthermore, it has a small footprint and a moderate power draw suitable for small aerial platforms. Overall, the Nano Jetson is presented as a practical and representative embedded computing solution for the proposed system to be tested on.

5.1.2 Software Stack and Development Tools

The thesis relies on the software stack and development tool set of choice for real-time multimedia processing, utilizing the embedded platform resources as much as possible and providing for dependable system development and deployment. The project relies on the Linux-based operating system, takes advantage of the open-source multimedia libraries, and incorporates standard remote-access solutions, providing for a transparent and reproducible development environment.

Operating System

Ubuntu Linux is the preferred OS, which is based on the embedded companion computer. Although other choices can also be offered, Ubuntu is a high-quality documented platform with a good support of multimedia, networking, and low-level control. Linux kernel and the device drivers make us able to interact with cameras, networks and storage equipments which we need to have in order to stream video continuously.

Linux is a powerful operating system that has sufficient process control and inter-process communication. In this scenario, the functions of Linux are involved in coordination of multimedia pathways and low-complex processes. Some of the utilities that exist between processes that enable real-time data exchange are named pipes and redirection to output; therefore, it is possible to include telemetry data in a video footage real-time. This is a very important necessity of a UAV since it has to be operable all the time. More so, the choice of Linux-based OS as a platform on which the system will be operated makes it compatible and easily debuggable and loggable, which is central to the project. As a matter of fact, all of the mentioned characteristics contribute to the stability of the system as well as simplify the development process.

Multimedia Frameworks

The system has multimedia capabilities that are founded on the GStreamer platform. Actually, GStreamer multimedia framework is founded on the concept of pipelines that transform the multimedia tasks of complex nature to directed graphs and modules to perform the tasks. A specific functionality is applied by each module or graph which includes video recording, format conversion, overlay drawings, and encoding, splitting, network sending, and disk writing.

Secondly, the pipeline design is manifested in the unified processing mode designed in this thesis. The whole video processing pipeline has been made into a single pipeline with GStreamer full; hence; reuse is simple and consistent when the output is redirected to other destinations. Meanwhile, buffering and timing are to a large extent under control due to the design of the pipeline.

The ability to exploit hardware-accelerated multimedia pipes is one of the greatest benefits of GStreamer in this respect. These pipes enable intricate functions like video compression be to be outsourced to special hardware. This improves real-time video compression and video streaming. Moreover, GStreamer has an extended architecture. This means that alterations and experimentation on the pipes would not cause a radical change in the setup.

Programming and Scripting Languages

The shell scripts are used as the major implementation of the brain of the control logic. Naturally this comes in very convenient when synchronizing Multimedia Pipelines and so on. All this is done through these scripts to bootstrap, tune runtime knobs and control jobs. This type of design is quite appropriate to an embedded system where there is low overhead and direct use of OS services is exploited. Telemetry is also handled by scripts to ensure that it is formatted in a manner that is friendly to the video stream. The telemetry data is constantly being decoded into text overlay data and sent together with the multimedia stream in real time. This enables

geotags to be dynamically updated without the need to streamline the stream.

The use of scripting languages introduces flexibility and capability to repeat. Should you desire to add another functionality or a new feature, you just change the scripts without necessarily re-compiling and it is also easier to test and debug. As a matter of fact, by following the normal scripting tools, it increases portability even in embedded platforms based on Linux.

Besides the system scripts, the Ground Control Station browser interface is developed with the following web technologies: The layout of the web page is presented in terms of HTML, and JavaScript is employed in order to create the rules or behavior concerning the processing of events and dynamic chat with the live stream server. The JavaScript also has the role of setting up and maintaining the WebRTC connection and real-time updating of the display of the videos. All the web tools and technologies make the live videos watchable in any platform through the Ground Control Station as it supports any modern web browser.

Remote Access and File Management Tools

The developers access remotely the embedded system in the development process, testing and evaluation stages using the Secure Shell (SSH) protocol. SSH will also give secure command-line access, which means that you will be able to start and stop multimedia pipelines, and tune settings and keep an eye on its actions without necessarily having to touch the UAV. This kind of remote control is very vital in the operation of UAVs, as it may not be easy to reach hardware parts.

WinSCP is used in file transportation and control where communication between an embedded system and a development station takes place in case of the secure data. WinSCP has the ability to transport source code files, configuration and videos taken of the system when it is operating. WinSCP is compatible with multimedia files and it has a graphical user interface which can be used to manage files.

The combination of SSH and WinSCP will provide a smooth process of development and maintenance. SSH and WinSCP will enable rapid update deployment, download of experimental outcomes, and help analyze logs of the system, which will help make the development process more effective.

5.2 Camera Input and Preprocessing

The camera input and preprocessing section is the first point of the video processing pipeline and it has the task of capturing the raw video data of the on board camera and converting it to a form that can be further encoded, streamed, and recorded. This step is very essential in ensuring that the compatibility of the video stream

with downstream pipeline components is achieved without compromising the real-time performance of the embedded UAV platform.

The system is connected to a camera device which is in turn connected to the companion computer that supplies the system with a constantly changing stream of raw video frames. The camera has a fixed resolution and frame rate that is appropriate in the aerial video real time and is balanced because the visual quality and processing needs of the camera. The camera hardware captures video frames in a compressed image format, which is compatible with the camera hardware, which lowers the data rate of the captured image and minimizes the bandwidth consumption on internal system buses.

After this, the camera output is deciphered and converted into a raw video signal which can be processed further in the multimedia pipeline. This step of decoding is necessary to make sure that further processing components work with a uniform and standardized video representation. Conversion of format to that of the color space and memory layout anticipated by a hardware-accelerated encoder in the latter stages of the pipeline is done. This preprocessing will be necessary to prevent format conversions that might be avoidable downstream and that may otherwise encourage more latency and computation.

Besides conversion of format, the preprocessing phase offers a logical point of visual insertion of visual enhancements and metadata overlay. Timestamps are drawn on the video frames, to offer time reference when viewing the live surveillance and after mission analysis. This timestamp overlay provides every frame with unique capture time that is especially useful in matching video data with telemetry data or other sensor values.

Integrating the geotagging overlays mentioned in Section 4.4 is also the task of the preprocessing stage. The information obtained via telemetry including the geographic position and vehicle attitude are represented in the form of textual overlay on the video frames. The system ensures that the live video and the recorded one have the same visual annotations by doing this integration prior to video encoding. This will make it easier to synchronize and capture that contextual information despite having watched the video without the UAV system.

Implementation wise, the operations in the preprocess are implemented to be lightweight and deterministic so that real time performance is not affected. Elements in the processing path are positioned in such a way as to reduce buffering and memory duplication and permit frames to pass efficiently through the pipeline. The sequence of decoding, conversion and overlay rendering is very selectively adopted so as to sustain compatibility with hardware-accelerated encoding and to evade the unnecessary incurring of multiple processing steps.

In general, the camera input and preprocessing step defines a consistent and constant video stream on which the further processing of the encoding, bifurcation of a video pipeline, streaming and record processes are based. The system guarantees consistency in video content in all outputs and low latency and resource consumption by decoding, converting formats, timestamping, and geotagging at the early stage.

5.3 Hardware-Accelerated H.264 Encoding

Once the camera has been fed and preprocessed, the video stream is sent to the encoding phase, where the raw video frames are compressed to lower the data rate and make transmission and storage of the raw video and other video data more efficient and at a reduced rate. The proposed system uses hardware-accelerated H.264 encoding in order to address the very high latency and performance demands of autonomous UAV applications with a resource-constrained embedded platform.

H.264 video coding standard is chosen because it is the most common, highly compression and capable of being used with both streaming protocols, web-based clients and media containers. It is more specifically applicable in the real-time aerial video streaming and recording due to its tradeoff between compression capability and computational requirement. By employing H.264, the live video stream may be provided with high reliability to the Ground Control Station and have the visual quality of moderate bitrates. In order to have real time performance, the system makes use of video encoding hardware, which is present in the embedded companion computer. The system also helps to take the encoding procedure off of the CPU by utilizing specialized hardware units, whereas software-based encoding methods use a large portion of the CPU load. The purpose of this is to enable the CPU to be used in other processes like telemetry processing, pipeline control, and system monitoring and therefore enhances system stability in general.

Implementation-wise, the encoding part is set to run to a fixed frame rate and a target bitrate that balances the quality of the video and the usage of the network bandwidth. To achieve low-latency operation the encoder is designed with minimum internal buffering and results in regular keyframes thereby enhancing stream robustness and decreasing start up latency on a live playback. These environments are specifically significant when it comes to a UAV scenario, where changes in scenes and dynamic movement are frequent. The encoding step is placed at the head-end of the video pipeline with all the output branches. This architectural choice will make sure that the video stream is only encoded once and that the same compressed stream is also used in live streaming as well as onboard recording. The system lowers the level of computation redundancy by not undertaking several encoding processes and ensures the uniformity of video representation in the various outputs.

Those video frames are captured by the camera in Motion JPEG (MJPEG) for-

mat and processed by extracting raw video frames with the help of a Hardware-accelerated decoder and decoding them. These decoded frames are subsequently run through a number of color space conversions to produce the downstream processing and hardware encoding compatible format. Before encoding, the timestamp data and the subtitle overlays based on telemetry are coded on top of the video frames, so that the contextual information always is embedded into the video itself.

Encoding step applies a hardware-based H.264 encoder which works on video frames held in accessible memory to the hardware. This encoder takes the intensive compression operation of the CPU and transfers it to the specialized multimedia hardware, which can dramatically reduce system load and allow operation in real time. The hardware acceleration is especially critical in application of the UAVs, where the companion computer has to process video all the time, and at the same time, it needs to handle the telemetry integration, network communications and system control activities.

The encoder will be set to operate at low latency, having a constant target bitrate and keyframe periodic insertion. Having a large number of parameter sets and keyframes makes streams more robust and decreases the time taken to recover in case of packets being lost during live transmission. They are necessary in the case of aerial video streaming, where dynamic and rapid movement is the rule, and the network conditions can change throughout the flight.

The architectural choice of the H.264 encoding stage being placed on an upstream position in the pipeline is an important one. This guarantees that the video stream is only encoded once and the resultant compressed stream of bits is used as a shared source by all downstream side products. The system reduces the number of computational steps, maintains a constant video quality, and increases the energy efficiency of the system by avoiding unnecessary encoding processes. Figure 5.2 is the complete pipeline generated after the implementation.

5.4 Pipeline Branching for Streaming and Recording

The video pipeline is split into simultaneous video streaming and recording into several output paths after the encoding of H.264 bitstreams and the decoding of bitstreams which is accelerated by the hardware. It is achieved by means of a pipeline branching mechanism, which enables a single stream of an encoded video to be broadcast to many consumers without replicating upstream processing steps.

The branching is accomplished with the help of a stream splitting element that copies the H.264 bitstream that has been coded into separate branches. The branches are separated by buffering queues so that the differences in speed of processing or network conditions in one of the branches do not affect the other. This separation

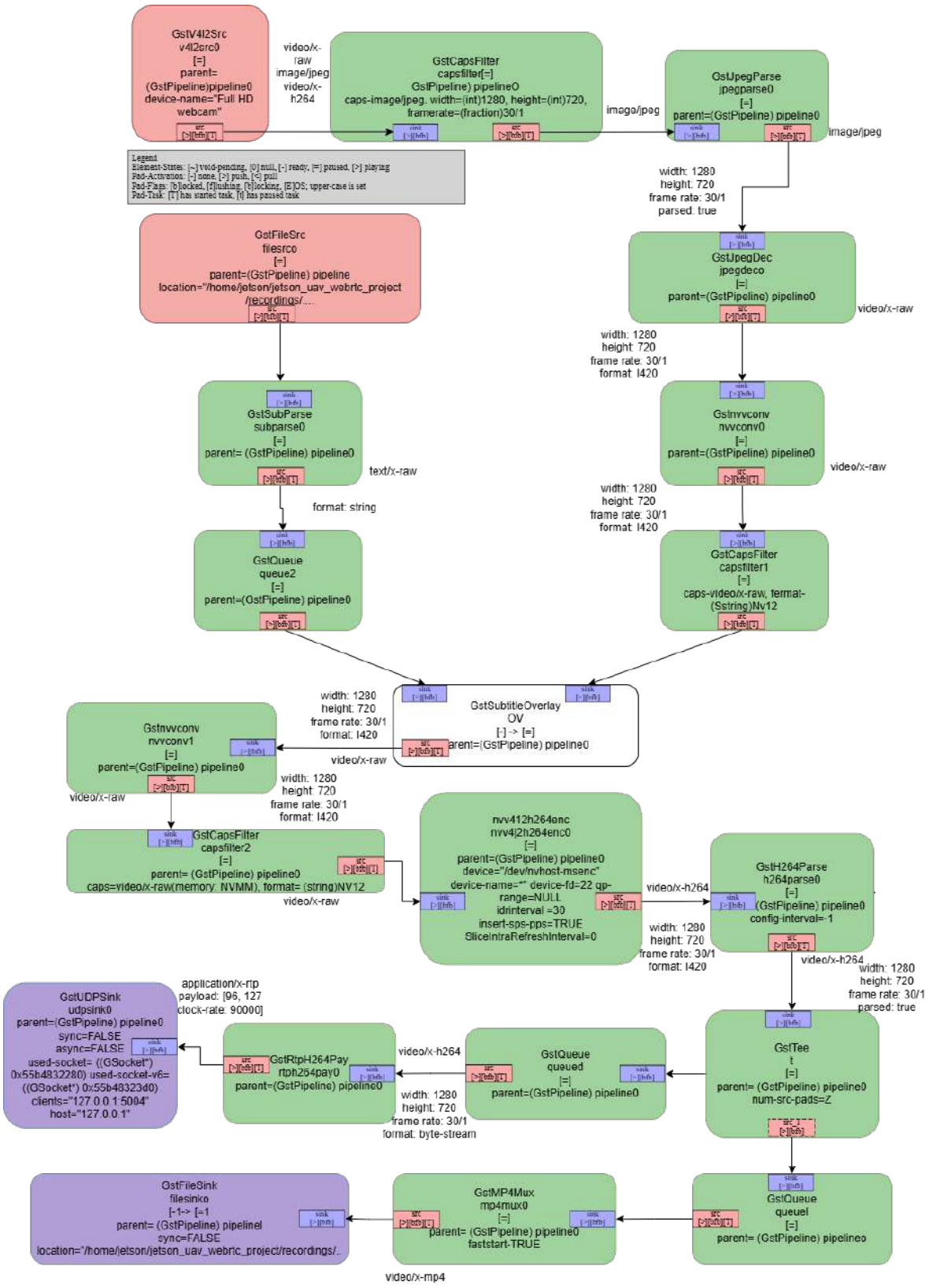


Figure 5.2: Generated GStreamer pipeline after implementation

is critical in keeping the real-time performance constant especially in cases where streaming and recording processes are characterized by different timing aspects. The pipeline has a branch that is devoted to real-time video transmission. The H.264 encoded stream is packetized in this branch into RTP packets that can be sent using the network. The UDP packetized stream is then sent over the low-latency connection to a media gateway which serves to make the live video accessible to the Ground Control Station. It is a branch that is geared towards minimal buffering and timely delivery, real-time situational awareness is more important than guaranteed delivery.

The second branch serves as a recording tap and takes the same encoded and packetized stream of video. This branch is intended to provide downstream recording or storage mechanisms without having an impact on the live stream. The system circumvents the extra encoding overhead by using the encoded stream instead of raw video frames so that the recorded data is temporarily synchronized with the live video.

There are a number of benefits of using a common encoded stream. First, it ensures that time has been made consistent between the live stream and the recorded video since both come out of the same compressed bitstream. Second, it enhances efficiency in the use of resources through the removal of unnecessary stages of processing. Third, it improves robustness of the system, since failures in a single output channel like network congestion or storage delays which do not cascade to other branches are avoided.

In general, the pipeline splitting approach makes it possible to apply video processing in a multi-output mode and be flexible and efficient in a single pipeline. With hardware assisted encoding, stream splitting and isolating branches, the system is capable of running both low-latency streaming and reliable recording at the same time, which is the essence of the intended UAV video system.

5.5 Network Streaming Implementation

The proposed system network streaming element is configured and provided with an aim of transferring live video on the Jetson nano on the UAV straight to Ground Control Station with as minimal a latency as achievable and satisfactory dependability to realize a live video viewing. To realize this functionality we use a protocol stack that uses both live video transport in combination with a live video solution that can be accessed by a browser. The entire live video transport chain is meant to be installed smoothly into the single video pipe, as well as latency constraints, which are to be followed during UAV operation.

Streaming is made up of two levels of processing. To begin with, the video, which has been encoded, is sent out of the Gstreamer processing system via real time and

low latency transport protocols. Lastly, the video is converted in a manner that it can be viewed directly through a standard web browser in the Ground Control Station without the need to have additional functionality provided through the extension of WebRTC connection.

Figure 5.2 shows the whole end to end implementation pipeline. The roles of the elements in the generated GStreamer pipeline are differentiated only through the application of different colors in the generated GStreamer pipeline overlay, as there is no visual clarity. Red elements often represent a pictogram of source as well as application-level components, the locations where data enter the pipeline. Green processing and transformation elements, like decoders, converters, encoders and parsers, indicate active stages of media manipulation. The sink elements, such as network and file outputs are presented in blue or purple, which is the last source of data output. The outgoing and incoming arrows of the blocks indicate the flow of data direction and the label of the arrows identifies the negotiated media format, resolution and frame rate respectively. Gstreamer does not standardize the color coding scheme but it is a way to help make sense of the pipeline structure and data flow.

5.5.1 RTP Streaming from GStreamer

During the final video processing pipeline stage the H.264-encoded video stream is divided into RTP packets. RTP packets are a real time audio/video protocol and give a nice sequence numbers and timestamps facility. This comes in handy in cases of dealing with video streaming.

These RTP packets are delivered through UDP and this minimizes latency. UDP eliminates the idea of retransmission, where there is a wait period associated with delivery of packages assured. This is not a loss in the situation of live video streaming, as the latest frames are received as soon as the package is received, and new frames are viewed as more valuable than late ones. It is corresponding with the concept of the UAV in which new frames are superior to perfect packages reception.

RTPoverUDP combined with GStreamerPipeline allowed delivering the stream of vital video information and having little overhead. This RTP/UDP data path can be defined as the handshake between the embedded processing and media server.

5.5.2 Janus Gateway Integration

The proposed system implements the Janus Gateway to be a WebRTC media gateway to allow a real-time delivery of video to a browser-based Ground Control Station (GCS). Janus is a mediator between the onboard video processing pipeline and web-based clients, which means that low-latency RTP video streams, created by GStreamer can be consumed by the WebRTC on a standard web browser. This

section outlines the role that Janus would play in the system and details the implementation of the integration.

Janus Gateway Overview

Janus Gateway Janus Gateway is an open-source general-purpose WebRTC server to facilitate real-time audio and video calls using web browsers. In contrast to monolithic media servers, Janus is based on a lightweight and modular design with the basic WebRTC implementation being augmented by a collection of independent plugins. Each of the plugins provides a single media handling or signaling behavior, making Janus able to be configured to suit a large variety of real-time communication situations.

Under the system architecture, Janus is used as a WebRTC end of life component, which handles signaling, session negotiation, and secure media transport between WebRTC clients and external media sources. It is not a rigid system of media processing but rather permits external media streams to be injected in WebRTC connections. The design of Janus makes it especially well suited to be used hand-in-hand with embedded systems and custom multi-media pipelines, where video capture and video encoding are not processed within WebRTC. It also embraced common WebRTC features like Session Description Protocol (SDP) negotiation, Interactive Connectivity Establishment (ICE) and secure media transport that allow it to work reliably in a heterogeneous network environment. Simultaneously, its simple design does not include any redundant media processing, therefore, the latency and the computational overhead is kept minimal.

Janus is a good solution in the context of the UAV video systems to bridge low-level video pipelines, which rely on RTP, and browser-based Ground Control Stations. Its plug-in structure enables the UAV systems to open the live video stream to the web clients without re-configuring the video processing pipeline. This feature makes Janus highly suitable when it comes to real-time UAV applications with low latency, modularity, and a low-deployment requirement.

Role of Janus as a WebRTC Media Gateway

Janus is an open, lightweight, modular WebRTC server that allows to bridge media sources not based on WebRTC with WebRTC based clients. In the suggested system, video capture and video encoding are not done by Janus. Rather, it accepts pre-dominated video circulations of the onboard GStreamer pipeline and directs it to linked WebRTC customers.

This design decision leaves the computationally expensive work in the development of the video capture, video preprocessing, video encoding, and video overlaying

to the Jetson Nano. Janus concentrates only on signaling, session control and protocol translation. The system no longer requires WebRTC processing at the point of delivering a media stream to the server, meaning it is modular and scalable and reduces the extra processing load on the embedded platform.

RTP to WebRTC Conversion

The GStreamer pipeline that will be used in the implemented solution is set up to generate an encoding video stream with the H.264 codec on the Jetson Nano. Once the video frames have been compressed the resultant coded video frames are then packetized into RTP packets and sent out using UDP. This packetization is done on the GStreamer pipeline with the help of RTP payloader elements which wrap up the coded frames along with the timing and sequence data used to deliver in real time.

The RTP stream is sent to one of the local UDP ports which are specially set to be observed by the Janus Gateway. This type of handoff between port and onboard video processing pipeline and WebRTC delivery subsystem is created. The point of RTP over UDP allows the system not to incur connection-oriented overhead and retransmission delays, and thus the low end-to-end latency is maintained.

When Janus receives the packets of the RTP stream, it consumes and links it to a WebRTC session. On the inside, Janus identifies the incoming RTP stream as a WebRTC PeerConnection by using matching codec parameters like payload type, clock rate and encoding format. Janus does not do any transcoding or re-encoding since the video is already coded by H.264 - a codec that is supported by WebRTC. The RTP packets are instead made to pass into WebRTC media pipeline.

This direct RTP to WebRTC forwarding route is one of the design decisions in the proposed system. It does not add a lot of computational load to the embedded platform, and it does not add any extra latency that would otherwise be caused by decoding and re-encoding. Due to this fact, the live video stream sent to Ground Control Station is highly similar in terms of timing and quality to the original stream that is encoded by GStreamer.

Janus Streaming Plugin Configuration

The GStreamer-Janus integration is based on the Janus streaming plugin that is specifically made to bring to the WebRTC clients the externally generated RTP streams. Within the suggested system, the streaming plug will be configured by a specific configuration file that will specify the characteristics of the received RTP stream.

The configuration includes parameters like the type of video codec (H.264), the payload type of RTP, network interface and the UDP port number, on which Janus

waits to receive the packets of RTP. These parameters can be explicitly defined, and then only by means of them will Janus correctly interpret the incoming RTP stream and identify it with a named streaming resource.

On start up of the system, Janus performs loading of the streaming plugin configuration and streams the RTP stream as a media source. Notably, such registration will take place without any client affiliations. Consequently, the GStreamer pipeline is able to start streaming RTP packets immediately after being started, whether or not a Ground Control Station client is already connected.

As a client who is a browser then connects to Janus with a request to the requested stream, the streaming plugin dynamically creates a WebRTC session and starts sending the received RTP packets to the client. This decoupled model of initialization is intended to make the video pipeline run continuously and not rely on the time of connection of clients, which is especially relevant when using UAVs since intermittent connectivity can occur.

Interaction with the GStreamer Pipeline

GStreamer pipeline and Janus Gateway interaction is based on a unidirectional RTP-over-UDP model of communication. After starting the GStreamer pipeline, it will keep relaying the encoded RTP stream to the UDP port that is set to Janus. There are no messages of signaling or control between GStreamer and Janus that occur in normal operation.

This weak interconnection of the two components makes the system architecture less complex. GStreamer performs no other functions other than media production, such as video capture, preprocessing, overlaying, encoding, and RTP packetization. In its turn, Janus deals with signaling, session negotiation, and WebRTC media forwarding. All the components act on their own and only communicate via the RTP stream.

This is better separation of concern, it enhances fault tolerance and robustness. In case of disconnection of the browser client, the GStreamer pipeline will stream data in the form of RTP packets without stopping. Likewise, when the video capture and encoding process is restarted, when the video capture/4=0 and the video encoding/0 is restarted as well, no restart of the video capture and encoding process is necessary. As soon as Janus becomes operational, it may start absorbing the RTP stream again. This option of design contributes to the stability of the system and enables the media generation and media delivery parts to be controlled independently.

5.5.3 WebRTC Streaming to Ground Control Station

The last stage of streaming configuration in the network is the delivery of the live video provided by the UAV to the Ground Control Station (GCS) with the WebRTC. To make the system user-friendly to the operator, the answer is the development of a web based client which incorporates the Janus demo GUI with a control page which was designed with the purpose of providing the required support.

The Web UI is created with usual Web technologies, HTML to organize, client-side scripting with JavaScript and designs with CSS. We do not want to write our WebRTC client on our own, so we chose the Janus demo site, where simple signaling and processing are already provided. The code of the demo page is increased and integrated into a custom page with only the necessary features being displayed leaving no superfluous information on the eyes of the operator.

The customized webpage will act as the main interface of the Ground Control Station. The video player is embedded in the webpage and further extended with the functions of the controls by the Janus WebRTC client. The controls that have been incorporated and used with the system are that of activation of the live video stream, activation of onboard recording of the UAV, termination of the recordings and, lastly, of termination of the live video stream.

The webpage controls the interaction with users and communicates with the Janus Gateway and UAV control systems. When a user decides to open a stream, it would connect to Janus through WebRTC, which would subscribe to the live video feed of the UAV. The recording operations are then converted into a command that changes the recording stream of the combined video graph without interruptions to the stream. When integrating the Janus demo functionality and the tailor-crafted web user interface, the system exploits the good WebRTC elements in addition to bringing in application aspects. This keeps the process of the development phase simple and the solution cannot be complicated to UAV operator by the bad interface. The system operates on the browser and thus does not need any dedicated client software to access the live video feed, which means that the UAV operator does not need any specialized systems to be operational.

In general, Ground Control Station web application is a useful destination point of WebRTC streaming and it unites live video processing and control of streaming and recording. It finishes a live streaming chain of the video processing chain on board and between Janus and the control station that is introduced at the operator level.

WebRTC Session Establishment

Upon a request transmitted to the browser-based Ground Control Station to open the live video stream, Janus starts the process of opening a WebRTC session. This starts by establishing a WebRTC PeerConnection that acts as a conceptual point of real-time media that occurs between Janus and the browser client. Janus creates an offer in the form of a Session Description Protocol (SDP) which specifies the media parameters it can support on the server, such as the type of codec, the format of the payload, and the transport properties.

The browser gives an SDP response, which confirms the similarity of media parameters, and finalizes the negotiation of the media session. In this stage, the network connectivity data is shared in order to find the most appropriate route between Janus and the browser. This negotiation makes sure that the connection between the result is secure as well as low-latency optimized ensuring that the real-time UAV video monitoring is possible.

When the WebRTC Peerconnection is correctly set up, Janus binds the incoming stream of RTP, taken into it through the GStreamer pipeline using UDP, to the negotiated WebRTC media track. Janus does not do any re-transcoding and re-encoding since the video stream is already in a WebRTC-compatible form of H.264. This direct forwarding scheme greatly minimizes the processing overhead as well as the end to end latency and maintains the real-time nature of the video stream.

Figures 5.3 depict the Ground Control Station interface that is browser-based and which is employed in the control and management of the live video streaming and onboard recording capabilities of the UAV system. Figure X represents the first or dormant position of the interface where the operator has access to control buttons to start and stop streaming and recording, to show and hide subtitles as well as the logs but there is no video on display. The WebRTC video area in this state is not active until the process of streaming is initiated by the operator specifically.

The active streaming state of the interface is shown in figure Y. Once the operator clicks the Start Streaming button, the Janus Gateway will create a WebRTC session with the browser and will then start relaying the RTP video stream that is created by the GStreamer pipeline. Consequently, the live video feed is seen on the browser. This is what indicates that video rendering in the Ground Control Station is event-based and relies on user interaction to guarantee that there is controlled access to live video streams. The two figures can be used together to visually verify that the GStreamer pipeline, Janus WebRTC gateway and browser-based Ground Control Station have been correctly integrated.

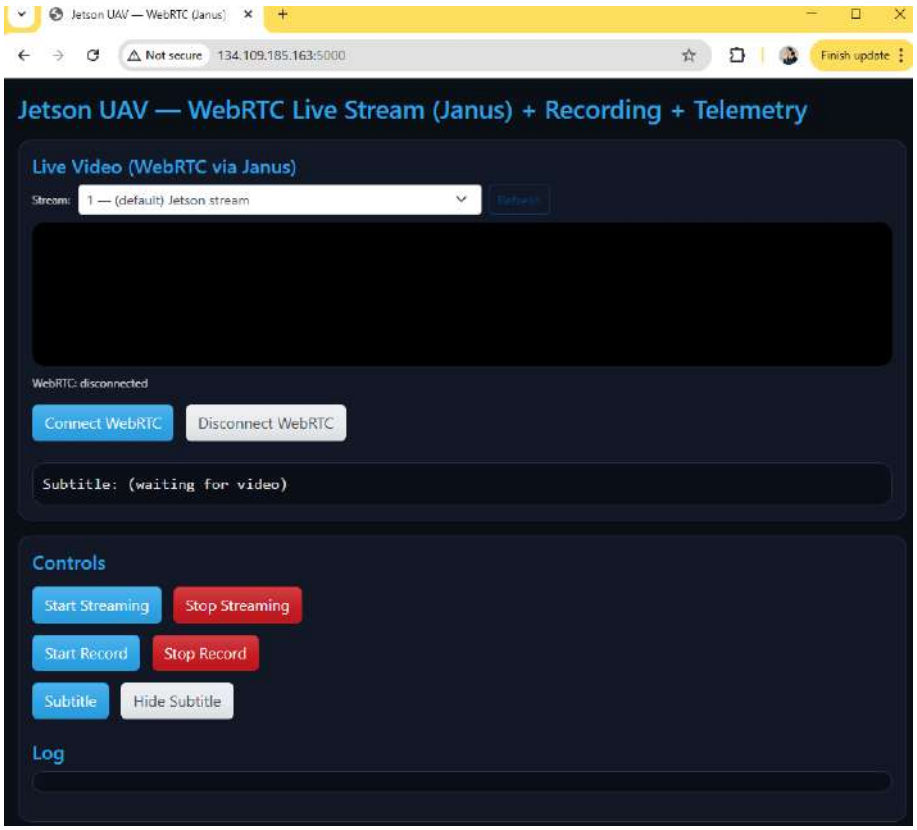


Figure 5.3: The browser-based Ground Control Station interface in idle

Video Rendering in the Browser

Once the WebRTC-session is established, the browser starts receiving the live video stream via the WebRTC media channel. The browser decodes the video frames it receives with its inbuilt WebRTC decoding support which is also designed to support real time media playback. This operation is carried out on the client side and consumes no extra complications or external software.

The video frames that are decoded are displayed directly through the web-based Ground Control Station interface. Because the embedded geotagging overlays are already embedded therein into the video stream inserted previously in the GStreamer pipeline, the video being displayed also contains all the associated contextual information including geographic coordinates, elevation, time and car orientation. This means that the browser does not need any further telemetry processing and visualization logic.

This design also makes the implementation of the design simpler on the client-side and guarantees that it is consistently visualized across various platforms and devices. The live UAV video can be accessed with the common web browsers, which

can be advantageous in terms of real-time functionality as well as immediate access to mission-related contextual information to the operators. The browser rendering solution also enables the incorporation of user interface controls such that when operators are watching the live video stream they are able to interact with the system.

Continuous Streaming and Client Independence

One of the architectural aspects of the proposed system is the decoupling of onboard video pipeline and client side viewing process. On the Jetson Nano, the GStreamer pipeline is always running capturing, processing, encoding, and streaming video independent of the connectivity of the Ground Control Station. The unidirectional connection between GStreamer and Janus initiated by RTP-over-UDP transfer that is not required to rely on client-side signaling makes this behavior possible.

Onboard video pipeline is not affected by the disconnection, refresh, or reconnection of the browser. RTP stream generation and transmission to Janus is not stopped. Creating a new browser connection, Janus only takes a new WebRTC connection and starts to send the old RTP stream to the client. This design guarantees the system robustness and also prevents useless restarts of the video capture or video encoding part.

In the same way, the system offers dynamic control of onboard recording where there is an active streaming session. The ability to start and stop recording without interfering with video delivery to Ground Control Station can be used. Such decoupled operation enables the system to sustain continuous real time streaming and at the same time has the ability to flexibly control onboard storage which aids in efficient use of resources and continuation of operations.

5.6 Onboard Recording Implementation

Among the parts of the system that would be suggested is the video recording features that will record the video on the computer that will be with UAV as the process of the video stream in real-time is performed. This would guarantee the availability of the most valuable videos made to be clear of missions achieved to be saved even when the networks of live video continue to cancel each other as operations proceed. Insofar as its implementation is involved, the recording to the device is achievable via a stream-splitting method in which the video stream is split once the H.264 video encode hardware acceleration is completed. This is intuitively the same as to say that the video being transmitted in the live feed as H.264 also becomes the source of the video recorded. Thus, the system is self-synchronous because it does not need the extra encode.

One of the important design factors is recording or streaming. Live video streaming is not interrupted in any way to facilitate situational awareness at the Ground Control Station. The recording service is however, an on-demand service meaning that it may start and end without affecting the live video streaming process and it may not need the Multimedia pipeline to be initialized. This will enable the operators to selectively log the mission-critical events.

The recording path is charged with the responsibility of capturing the encoded video data into an MP4 file format which is the desired choice in this case as it is supported by most of the multimedia players, analysis programs and post processing software in the market. The merits of using MP4 file format are the ability not only to store very long records in a convenient way but also to play the files in another setting easily. The files stored are therefore able to be opened easily so as to be analyzed without the process of undergoing any form of decoding. They also have overlays; timelines and telemeters which have been imposed upon the videos, as an extra feature of the video files themselves. In the real processing of the videos, the overlays are put in and included in the files in the subtitle format. Therefore, their data is now stored separately in the file, unlike the need to interrupt during the process of synchronization to see the real data holding each file.

The recorded videos are stored in the local drive of the companion computer and could be reinstated once the mission is over, provided that the network connectivity is made, by moving the video to the Ground Control Station. As the overlay and encode operations apply to the video streaming and video recording, the images that are seen when the video is streaming, are the same images that are stored in the video file exactly as they are in it. In this pronouncement, the narrator emphasizes on the ability to store and get the video after the mission once the network is created. The recorder on board has no dependability on the streaming path in terms of resilience. The glitches in networking and loss of packets and even temporary failures in the live streaming will not disrupt the recording procedure and the mission information will be secure. This contributes to the credibility of the system as there is low latency in live streaming.

5.7 Implementation of Telemetry

The telemetry integration is a vital component of the proposed UAV video streaming and recording system because it will be used to retrieve the necessary background data that will be used to increase the interpretability and usefulness of aerial video data. In the present implementation, telemetry processing is purposely constructed as a separate but time-synchronized pipeline, which is running with the video processing pipeline. The rationale behind this architectural choice is that telemetry processing is decoupled both from video encoding and transport and yet succeeds

in being time-synced to live video streams and recorded video streams. The UAV has the sensors of the inertial measurement unit (IMU) and the Global Positioning System (GPS) as sources of telemetry data, supplying such parameters as the geographic position, altitude, orientation (roll, pitch, yaw), speed, and heading. The system also supports simulated telemetry sources that create time-varying values in a smooth manner so that the system can be tested and developed. All the telemetry data are structured as objects, regardless of origin, and are of the format of structured JSON objects, making them consistent and extensible. A telemetry packet contains the exact UTC time that is the common time in the system.

Telemetry information is sent to the Jetson platform in the User Datagram Protocol (UDP) over a special port. UDP is selected based on its low-latency property and minimal overhead that it has, and when application telemetry delivery requires real-time and consideration of instances of packet loss is not a critical factor in system operation, it is appropriate. The system also ensures that all the telemetry transport does not introduce any additional load to the video transport, ensuring that video streaming and recording processes remain deterministic.

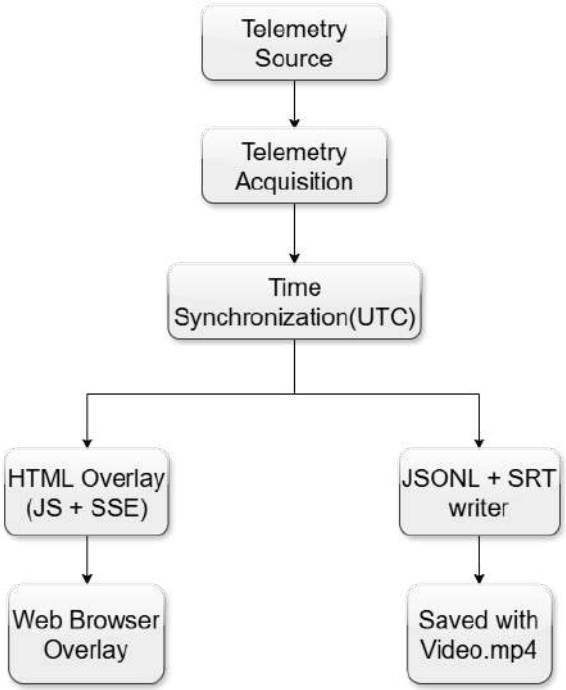


Figure 5.4: Implementation of geotaging.

5.7.1 Telemetry Acquisition and Parsing on Jetson

The telemetry acquisition within the proposed system is a software process that is a standalone operation on the Jetson board. This component receives telemetry data produced by the UAV navigation and sensing subsystems, decodes the data received, and transforms it into a formal structure that is taken inside the system as an internal representation. Telemetry information is sent using User Datagram Protocol (UDP) socket to a port located at a fixed position on the Jetson device. UDP is chosen because it has low overheads in communication and is suitable in real-time data communication whereby the latest telemetry data has a higher priority than ensuring each packet has been delivered.

A python based telemetry listener is used to constantly monitor the UDP port and handle packets as they come. The packets are decoded in a text representation and parsed as a JSON object. The JSON format allows the recognition of the separate telemetry variables like latitude, longitude, altitude, pitch, yaw, roll, speed, and heading. This structured representation can be easily processed in real-time and is easily stored over time and also can enable the easy expansion of telemetry fields in the event of any subsequent revision to the system. The telemetry values are interpreted into numeric formats where applicable and checked to be consistent once they have been parsed. The latest telemetry snapshot is saved in a common data format which is available to other system assets. This architecture will see to it that downstream consumers will always have a consistent and current view of the state of the UAV. When the telemetry updates cannot be received, the system will use the final sound snapshot that will ensure continuity in both the display and recording output.

Significantly, there is no relationship whatsoever between telemetry acquisition and parsing and the video processing pipeline. Telemetry information is not routed through GStreamer elements or can make any difference in video encoding or transport performance. This segregation boosts the resilience of the system and eliminates any telemetry processing that could add to the video stream in terms of latency and instability.

5.7.2 Time Synchronization Strategy

Time synchronisation becomes the key factor in the harmonisation of telemetry data with the related video material. Synchronizing and proposing system In the proposed system, a software-based timestamping plan will be used based on a common time reference on the Jetson platform. When the system receives a telemetry packet, it allocates a timestamp based on Jetson system clock which is set to work according to Coordinated Universal Time (UTC). This time is the exact time when the telemetry data is made available to the processing pipeline.

Video recording sessions start time are set using the same system clock reference. As the recording operation starts, the time at which that point is recorded is taken as recording start epoch. The time of all the telemetry measures taken in the same session is relative to this reference. This means that the timing of telemetry and video timelines has a common time base and through this, video frames and telemetry values can be properly correlated while viewing and analyzing their data.

In this strategy of synchronization, no hardware timestamping is required or direct interconnection of the telemetry packets and the video frames. Temporal alignment is rather pursued by using the same system time in independent pipelines. It is an appropriate methodology to software-defined UAV systems, where the accent is made on flexibility, portability, and integration simplicity. It can also be used to provide post-mission analysis to run telemetry data replayed and compared to particular locations in the recorded video based solely on timestamps.

5.7.3 Telemetry Distribution

Once telemetry data has been collected, processed and time-stamped, it is sent concurrently to two different processing streams, a live telemetry distribution stream and a recording telemetry stream. Both routes are based on the same synchronized telemetry snapshot, and the real-time visualization and the data will be equal. Server-Sent Events (SSE) is used to push the telemetry data to the web-based user interface in the live telemetry distribution path. The Jetson system is an SSE server that iteratively transmits telemetry messages to web clients that are attached to it. JavaScript logic on the client side is then updated with this and displays the telemetry data as an HTML overlay on the WebRTC video stream. This overlay is also dynamic and indicates the latest telemetry snapshot to allow operators to view the UAV state in real-time and the live video feed. The overlay is created at the browser level, which means that it does not deal with changing the underlying video stream, and presents no extra encoding burden.

Simultaneously, the recording telemetry route is a process that provides the continuous storage of telemetry information when recording video. Telemetry snapshots are written periodically when recording is enabled to two complementary file formats, a file containing system state as a JSON Lines (JSONL) file and one containing a transcript as a SubRip subtitle (SRT) file. The JSONL document saves the raw telemetry data in a machine readable format that can be used later in the data analysis, visualization, or integration of post mission telemetry data. The SRT file, in contrast, gives a time-stamped text record of telemetry data that may be replayed on the MP4 video recorded in regular media players.

Both the JSONL and SRT like are based on timestamps expressed with reference to the common starting point of recording, so that the reported time is accurately

correlated with the video content being recorded. This bi-format storage plan provides a balance between human readability and flexibility of analysis and does not imply the complexity of using embedded metadata standards. This means that the telemetry data and recorded video is self-contained, descranged and can be viewed with no special decoding software.

5.8 Robust Pipeline Control and Operational Reliability

Quality work in dynamic environments UAV-based video streaming systems require reliability due to frequent user-controlled actions, variability of the network, and long run-times. In order to compensate these difficulties, the suggested system is equipped with robustness-based design decisions and regulated pipeline operation tools, which assist in maintaining stable operation in the case of long-term missions.

This is done by explicit operator commands via a web-based interface which controls the startup and termination of the video processing pipeline. Onboard recording and streaming are independent and can be started and stopped without starting the whole pipeline. The safe and smooth closure with the end-of-stream processing is guaranteed by the active closing of the GStreamer pipeline and the ability to run buffered videos and properly end video files.

Functional decoupling is another way of improving system robustness. The live streaming, onboard recording, and generation of telemetry are enforced as autonomous subsystems that have a common upstream processing step. Subsequently, onboard recording is not affected by interruptions to the streaming subsystem including temporary network failures or browser loss of connection. Equally, video capture and encoding is not interrupted by the delay in telemetry updates.

The acquisition of telemetry is non-blocked through the integration of an external process that produces data on subtitles continuously. This information is fed to the video pipeline without interrupting the video processing. Without providing new telemetry, the video pipeline will keep running normally, providing continuous streaming and recording.

Telemetry generation necessitating auxiliary processes are checked upon system startup and started otherwise. This will enhance the reliability in operation without wasting time by restarting the pipeline. The system is made to be long-run, taking advantage of the hardware based video processing and not having to restart the pipeline too often keeping the resource usage steady.

To conclude, the proposed system can be said to have robustness in terms of the controlled pipeline management, decoupling of sub systems, and non-blocking telemetry integration. The design considerations allow to guarantee real-time streaming and onboard recording reliability at different operational conditions that usually arise during UAV deployments.

6 Results and Evaluation

6.1 Functional Evaluation

The functionality check of the proposed video streaming and recording system of the UAV is done to ensure that all the components employed are working properly in the usual conditions of operating the system. The main aim of this test is to prove that the system meets its functional requirements, such as the live video capture and processing, live video streaming to a browser-based Ground Control Station, on-demand onboard recording, and built in geotagging and telemetry overlays.

The initial validation of the proper operation of the GStreamer-based video processing pipeline is the validation of the successful video capture, preprocessing, encoding, and pipeline branching. The live streaming subsystem is tested on functional behavior through the next step of ensuring web to end video delivery is implemented using WebRTC. Active streaming session selective recording capability is then analyzed in order to ascertain the independent and reliable onboard recording. Lastly, the integration of the telemetry and the timestamp data in the streamed and recorded video outputs, proper functionality of the web-based control interface, etc. are assured.

The outcomes provided in this section form a checkpoint confirmation of system accuracy and robustness, which forms the required basis of the quantitative performance test given in subsequent sections.

6.1.1 Verification of the Video Processing Pipeline

The main part of the developed system is a multimedia pipeline based on GStreamer and running on NVIDIA Jetson Nano. The video acquisition of the pipeline starts with the usage of a USB camera with Video4Linux2 (V4L2) interface. The camera has a 1280x720 pixel Motion JPEG (MJPEG) video, which is captured at 30 frames per second. As MJPEG is a compressed format, the incoming frames are first decomposed and decoded with the aid of NVIDIA hardware-accelerated MJPEG decoder followed by conversion into a raw video format which can then be further handled.

At the preprocessing stage, the telemetry information produced by the UAV system such as timestamps and geospatial data are incorporated. A background telemetry process continuously writes telemetry values in a subtitle (SRT) file, and super-

imposes the values on the video frames in real time with a subtitle overlay element. This guarantees the encoding of geotagging information as an integral component of the visual information.

Designed to be used on the NVIDIA hardware encoder (NVENC), the video stream is preprocessed followed by overlay integration, followed by compression of the video stream. Hardware encoding is far more efficient in terms of CPU overhead as well as permitting real-time operation. The coded stream is then divided with a tee element into several branches whereby, the live streaming can be performed and onboard recording can be seen using one encoding stage, simultaneously.

One of the streams encodes the H.264 stream as RTP packets and transmits them over UDP to the Janus Gateway, and the second branch sends the encoded stream to be recorded. The two branches work simultaneously and autonomously, thus, there is no interference between live streaming and recording.

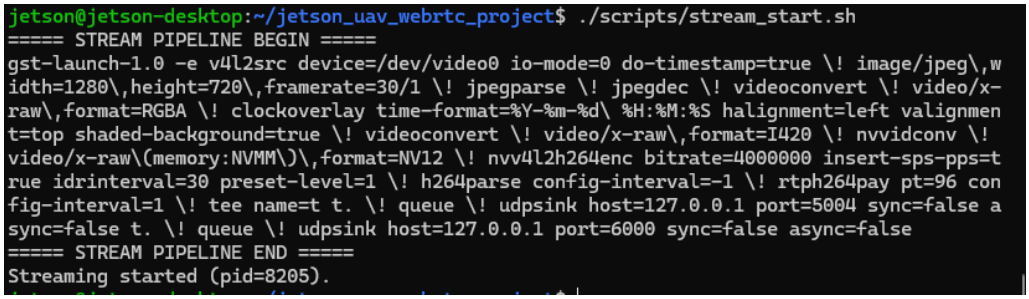


Figure 6.1: Pipeline startup and NVENC initialization log

The pipeline is initialized and works correctly due to the confirmation of the runtime logs and system observations. The pipeline as depicted in Figure 6.1 manages to move to the PLAYING state after moving out of the PAUSED state, which means that all the elements are properly configured and connected. The activation of the NVIDIA multimedia components and H.264 encoder is further confirmed by the log output, which proves that hardware acceleration is being used.

Process inspection is used to verify the active pipeline instance with the full GStreamer command line showing that preprocessing, overlaying integration, hardware encoding, tee-based stream splitting, and RTP transmission components all exist. This gives first-hand evidence that the planned pipeline is operating as desired. Lastly, the verification at the network level is done with the help of a packet capture on the UDP port that is used to setup RTP streaming. Various UDP packets are observed to have constant packet sizes and lack of packet drops caused by the kernel as indicated in Figure 6.2.

```
jetson@jetson-desktop:~/jetson_uav_webrtc_project$ sudo tcpdump -i any -n -vv udp port 5004
tcpdump: listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
19:49:06.099506 IP (tos 0x0, ttl 64, id 7529, offset 0, flags [DF], proto UDP (17), length 92)
    127.0.0.1.35250 > 127.0.0.1.5004: [bad udp cksum 0xfe5b -> 0x575e!] UDP, length 64
19:49:06.160581 IP (tos 0x0, ttl 64, id 7534, offset 0, flags [DF], proto UDP (17), length 50)
    127.0.0.1.35250 > 127.0.0.1.5004: [bad udp cksum 0xfe31 -> 0xa509!] UDP, length 22
19:49:06.160639 IP (tos 0x0, ttl 64, id 7536, offset 0, flags [DF], proto UDP (17), length 44)
    127.0.0.1.35250 > 127.0.0.1.5004: [bad udp cksum 0xfe2b -> 0x6dc4!] UDP, length 16
19:49:06.160680 IP (tos 0x0, ttl 64, id 7537, offset 0, flags [DF], proto UDP (17), length 50)
    127.0.0.1.35250 > 127.0.0.1.5004: [bad udp cksum 0xfe31 -> 0xa507!] UDP, length 22
19:49:06.161025 IP (tos 0x0, ttl 64, id 7545, offset 0, flags [DF], proto UDP (17), length 44)
    127.0.0.1.35250 > 127.0.0.1.5004: [bad udp cksum 0xfe2b -> 0x6dc2!] UDP, length 16
19:49:06.161073 IP (tos 0x0, ttl 64, id 7546, offset 0, flags [DF], proto UDP (17), length 50)
    127.0.0.1.35250 > 127.0.0.1.5004: [bad udp cksum 0xfe31 -> 0xa505!] UDP, length 22
19:49:06.161135 IP (tos 0x0, ttl 64, id 7547, offset 0, flags [DF], proto UDP (17), length 44)
    127.0.0.1.35250 > 127.0.0.1.5004: [bad udp cksum 0xfe2b -> 0x6dc0!] UDP, length 16
19:49:06.161429 IP (tos 0x0, ttl 64, id 7548, offset 0, flags [DF], proto UDP (17), length 1428)
    127.0.0.1.35250 > 127.0.0.1.5004: [bad udp cksum 0x0394 -> 0x1c1b!] UDP, length 1400
```

Figure 6.2: RTP packet capture using tcpdump

This proves that the RTP stream is being produced and sent on a continuous basis, which confirms that the streaming section of the pipeline is right. All these observations taken together prove that the video processing pipeline is fully functional, does the real-time capture, overlay, encoding and stream generation and gives a stable base to both live streaming and onboard recording.

6.1.2 Live Video Streaming Verification

Live video streaming functionality of the proposed system was functionally tested with the help of browser-based Ground Control Station (GCS) interface and runtime logs of the Janus Gateway and onboard Jetson Nano. The proper start of the WebRTC streaming backend is proven by the logs of Janus server. Logs show that the Janus streaming plug is loaded and initialized and that the transport is the HTTP. Such messages affirm that using the onboard system, Janus is properly set up and ready to consume RTP streams and open them to WebRTC clients untranscoded.

Figure 6.3 shows what happened to the Ground Control Station (GCS) web interface as soon as the operator clicks the Start Streaming button. This button in the present system activates the onboard GStreamer video pipeline on the Jetson platform by a REST API request. When a trigger is received, the camera feed is captured, processed, encoded and sent to the Janus WebRTC gateway. Consequently, the live video stream is visible in the browser interface at the desired resolution of 1280x720.

The WebRTC connection is already established with the help of the Control WebRTC. This is attested by the log panel on the right-hand side where messages like the one below are displayed: "Janus connected. Added streaming plugin attaching as well as Remote track: kind=video on=true. These log entries prove that the browser has negotiated a WebRTC session successfully with the Janus server and it is now preparing to accept media when streaming is started. The fact that the live

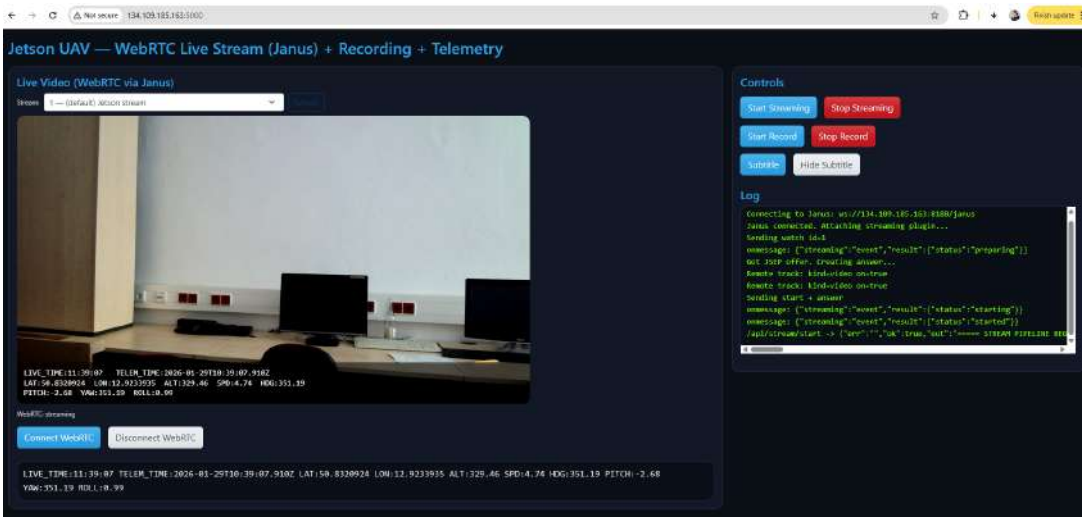


Figure 6.3: WebRTC video stream live in the browser following the activation of the Start Streaming.

video feed is not provided until one presses Start Streaming proves that the media pipeline is not in a running state but is provided on demand by the operator.

Besides the video, there is a telemetry overlay that is presented at the top of the live stream. Such overlay also contains real-time data in the form of UTC timestamps, latitude, longitude, altitude, speed, heading, and attitude (pitch, yaw and roll). The fact that this data is being received independently and that the Python telemetry service is being used to process this independent data, that time-synching has taken place, and that data has been rendered in the browser by using Server-Sent Events (SSE) substantiates this point. Log messages reported on telemetry updates also substantiate the fact that telemetry distribution is working and in sync with video playback.

The interface state is depicted in figure 6.4 following the operator press desecating the Hide Subtitle button. This has an influence on the client-side visualization of telemetry only. After activation, the telemetry overlay is dropped off of the video display and the subtitle status is changed to hide, as indicated below the video panel. Notably, the video stream is not interrupted in any way and telemetry is being processed and stored in the background. This shows that the telemetry visualization can be decoupled with video capture and telemetry acquisition and offers the flexibility of user control without affecting the operations of the core system.

Stop streaming button will stop the onboard GStreamer streaming pipeline leaving the WebRTC connection. This behavior is indicated by the log output where it is indicated that the stream has been stopped and still the Janus session is active. Consequently, the video screen is frozen or disappears but the browser is open

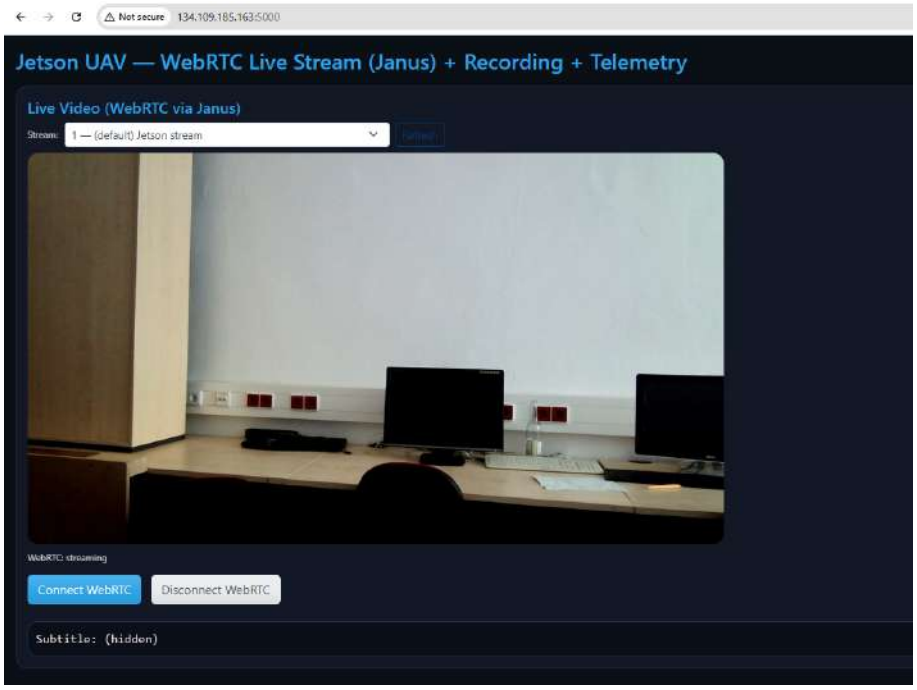


Figure 6.4: State of browser interface on selecting Hide Subtitle.

and connected to restart a stream should the operator presses Start Streaming once more. This proves that the system is capable of dynamic start-stop enabling video transmission without the need of complete reconnection.

6.1.3 Onboard Recording Verification

The video recording capability was tested to confirm that the system is capable of recording mission-relevant video along with its related telemetry reliably without interfering with the live streaming. The recording mechanism is supposed to be selective, operator-controlled so that only when needed, it can be stored. This is especially applicable to UAV missions when there is no need to record continuously and onboard storage capacity has to be spared.

The Ground Control Station (GCS) interface can be used to control recording via dedicated Start Record and Stop Record controls and is web-based. After clicking the Start Record button, the already running H.264-encoded video data stream is kept running through the transmission of the stream to the browser through the WebRTC, and the parallel recording process on the Jetson platform is implemented. The video stream is diverted to the recording branch and recorded on the local disk in an MP4 container without affecting the live stream. Meanwhile, telemetry acquisition is started and received telemetry data are incessantly time-stamped and logged to neighboring SRT subtitle files as well as JSONL telemetry logs in the same

recording directory. This makes every video piece recorded always to be linked with the telemetry data.

When the Stop Record control is activated, then the recording of video is gracefully ended and the MP4 container is completed to maintain its integrity. At the same time, logging telemetry is also ceased under the controlled nature, such that, subtitle and telemetry files are fully formed and aligned properly with the video that is recorded. Notably, pausing the recording is irrelevant to the live WebRTC stream, which proceeds with a normal operation and the operator can restart recording at any point of time in the mission.

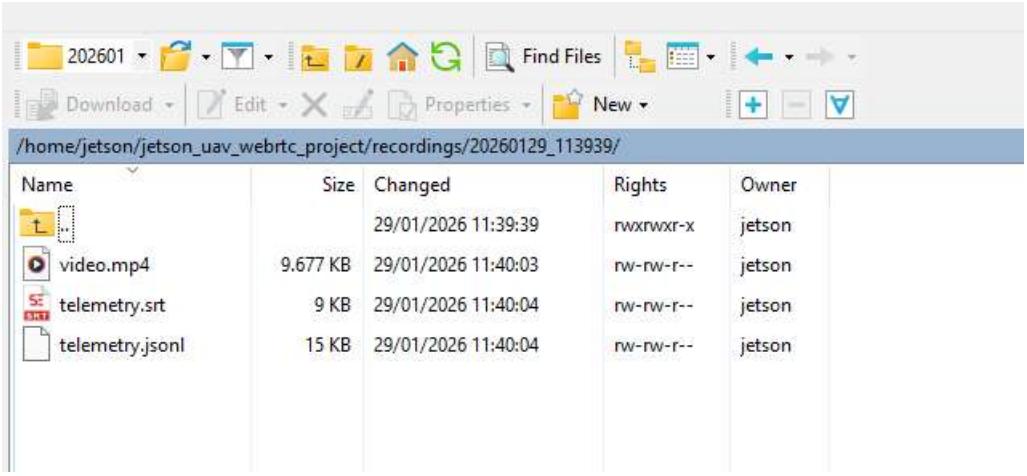


Figure 6.5: Video files recorded on the Jetson Nano as MP4 files.

The video files and telemetry files will be recorded in a specified recordings directory on the Jetson Nano as shown in Figure 6.5. Folders with timestamps are created in multiple folders, in accordance with a start and stop actions initiated by the user. The existence of the valid MP4 files, and the corresponding SRT files and the JSON telemetry files, with appropriate file sizes and modification dates, proves that every recording session is completed successfully and closed correctly.

The playback tests involving external systems ensured that the recorded MP4 files are sound, properly encoded and not corrupt. In playback mode, it is possible to see the same telemetry data as can be seen on the live stream display, i.e. timestamps, geographic position, altitude and vehicle attitude via the subtitle overlays. This shows that the telemetry information is synchronized with the video material and stored with it to be analyzed later during mission.

It is worth noting that, recording starts or stops do not disrupt the live video broadcast. WebRTC stream is continuous regardless of recording state changes, which is a confirmation that the system architecture can effectively separate live

streaming and recording functions without losing the synchronization. This supports the usability of the proposed system in the real-world UAV applications where the operator might require the ability to selectively record vital mission footage and related telemetry without interference with real-time surveillance.

6.1.4 Geotagging and Telemetry Overlay Verification

Geotagging capability was also tested to confirm that telemetry information, live and recorded video streams were obtained, synchronized, and displayed correctly. The proposed system also receives telemetry data (data about the time, spatial position, altitude, and vehicle attitude) without being filtered through the video pipeline and acts on the Jetson platform. It uses this telemetry data to produce real-time visual subtitles to be viewed live and synchronized subtitle and log files at the time of recording.

When used in live, the WebRTC video shown in the browser will show an incoming stream of telemetry information in the form of a readable overlay. The timestamps displayed in the overlay are real time and are synchronized with the system clock and indicate that the telemetry updates are handled and sent without any observable latency. This validates real-time association between the received telemetry data and the visualized video frame in spite of the video and the telemetry being carried by separate channels.

The corresponding telemetry is stored when recorded MP4 files are replayed either as a synchronized subtitle (SRT) and structured JSON log files created during the recording session. The entries in the subtitle have been made at the time of the video recordings and this guarantees that a similar mission-relevant telemetry situation that is present on the screen during live operation can be examined after the mission.

The system has the benefit of keeping both the live streaming and recorded outputs self-contained in terms of contextual information by separating the telemetry handling and the video transport yet having a common time reference. The overlay format and update rate were created in such a way that the important visual information is not hidden to ensure that it can be read and it can be viewed in real time as well as offline.

Altogether, the findings prove that the introduced system is stable to combine synchronized geotagging data and UAV video outputs. Incorporation of Live Telemetry overlays to WebRTC viewing in conjunction with synchronized telemetry files to recorded sessions meet the functional needs of the UAV video systems offering a realistic balance in usability of the systems in real-time, the ability to analyze the session post-mission, and the complexity of the systems.

6.2 Quantitative Performance Evaluation

After the successful functional testing of the proposed UAV video streaming and recording system, this section provides a quantitative analysis of the performance of the same. This evaluation aims at evaluating whether the implemented architecture can be used to meet real time operational requirements when deployed on an embedded UAV platform. The quantitative evaluation also places less emphasis on usability and mission effectiveness as compared to the functional evaluation which evaluates the aspect of behavior in terms of correctness.

The performance measurements will focus on three areas: the end-to-end latency of the live video stream, the stability of the streaming and frame continuity in the working conditions, and the use of resources on the Jetson Nano platform. The metrics are important in real time applications of the UAV, as delays in the visual feedback, unstable streaming, or excessive computation can be detrimental to the operator decision-making and the overall reliability of the system.

Each and every measurement is taken when the system is working in its full-integrated state, and on-demand live WebRTC streaming and onboard recording are turned on. This makes sure that the stated results are realistic in terms of deployment.

6.2.1 Resource Utilization Analysis

Resource utilization the ratio between the available system computational and memory resources and the workload done by the system. Resource use is a key performance indicator in embedded real-time video processing systems, e.g. unmanned aerial vehicle (UAV) video streaming platforms. Efficient use would make sure that the system is able to operate continuously, allow real-time constraints and even future functionalities without affecting performance. On the other hand, overuse can lead to the rise of latency, lost frames, thermal throttling or instability of the system. Thus, the CPU, GPU, and RAM utilization will have to be studied thoroughly to assess the efficiency and scalability of the offered video streaming, video recording, and telemetry architecture.

In order to measure system resource utilization accurately, tegrastats utility by NVIDIA was used in all experiments, which is a lightweight, real-time monitoring utility, unique to NVIDIA Jetson platforms, and it reports periodically CPU load, GPU utilization, and memory consumption with low overhead. This gives it good compatibility with performance profiling of embedded systems where intrusive profiling tools can introduce undesirable effects in real-time behavior.

In both experiments, the tegrastats was constantly running in the background as the video pipeline based on GStreamer was active. The tool created a record of

all timestamps of CPU use in all cores, GPU use, and RAM use at specified time intervals. These logs were diverted to files and eventually processed offline in order to extract quantitative metrics. This will make sure that the reported measurements are a measure of actual runtime behavior and not a snapshot of the behavior.

Experimental Setup and Test Case Execution

In order to have accurate and representative measurements a period of 10 minutes was carried out in each test case. This wait time was selected to ensure that the system had stabilized and to reduce the impact of the short lived transient effects like the initiation of the pipeline, cache warming and allocation of buffers. Executing each setup over a long duration also allows monitoring any spikes in the consumption of resources that are significant in determining the robustness of the system.

In both cases, the tegrastats logs were recorded and both the mean and peak values of CPU utilization, GPU utilization and RAM consumption were calculated. Mean values indicate the demand on resources when the system is operating at normal performance whereas maximum values characterize worst-case behavior, especially when critical performance is required by the system in real time.

The tests were performed at two resolutions of the videos **1280x720 (720p30)** and **1920x1080 (1080p30)** and three different working modes:

1. **Streaming only**, in which real video is served using WebRTC.
2. **Streaming along with recording**, during which the video stream is being recorded to file.
3. **Streaming with recording and telemetry(Geotagging)** This is where the video streaming and recording are coupled with real-time telemetry data processing.

With this arrangement it is possible to perform a systematic study of the impact of the increase of functional complexity and video resolution on the use of system resources.

Quantitative Resource Utilization Results

Table 6.1 shows the average and peak CPU utilization, average and peak GPU utilization and average and peak RAM utilization of all of the six test cases. In the streaming-only case, where the 720p30 system is used, the system has an average CPU load of 27.57% with a peak of 53.50%. The peak value is relatively higher and this is attributed to the short term events like the startup and network I/O operations of the pipeline. The usage of GPUs is low (4.72%) and means that the hardware process of encoding takes place efficiently, and the average amount of RAM used is 1812.8 MB that represents a constant memory footprint during steady streaming.

Test case	CPU Usage (%)		GPU Usage (%)	RAM Usage (MB)	
	Avg.	Peak	Avg.	Avg.	Peak
720p30 – Streaming	27.57	53.50	4.72	1812.8	1815
720p30 – Stream + Record	27.44	29.50	4.75	1814.4	1817
720p30 – Stream + Record + Telemetry	27.53	29.75	3.84	1840.4	1843
1080p30 – Streaming	26.36	32.00	5.26	1813.1	1814
1080p30 – Stream + Record	26.63	28.50	6.22	1816.1	1819
1080p30 – Stream + Record + Telemetry	26.86	50.75	6.34	1853.7	1857

Table 6.1: Average and peak CPU, GPU, and RAM usage for different operating modes and video resolutions.

When it is recorded at 720p30, the average CPU usage is almost the same at 27.44% and the maximal CPU usage drops to 29.50% implying more stable load distribution in the sustained operation. The usage of GPU is not much higher (4.75 %) and the consumption of RAM is growing a bit more (1814.4 MB) as extra buffering and file I/O traffic is related to recording.

Adding telemetry at 720p30, the CPU usage is nearly the same, with 27.53%, and the GPU usage is slightly lowered to 3.84%, because telemetry processing is rather CPU- and memory-intensive. Nevertheless, average RAM consumption is raised to 1840.4 MB, and the highest value is 1843 MB as the memory consumption is overhead with telemetry data structures and synchronization.

At 1080p30, the same tendencies can be observed, although resolution is the primary factor with which to influence the use of the GPU. During the 1080p30 streaming-only scenario, we see that the average CPU consumption is 26.36% and that the average use in the case of the graphics card is 5.26% since the more intensive video frame encoding requires more computation. The amount of RAM used is constant at 1813.1 MB which means that the resolution changes alone do not have much effect on the memory usage.

In the case of 1080p30 recording, the average use of CPU 26.63% and GPU use is even higher 6.22% and the use of RAM is also slightly higher 1816.1 MB. The 1080p30 stream + record + telemetry configuration indicates the greatest total use of the resources with an overall average of 6.34% of the GPU used and 1853.7 MB average of the RAM used with the highest value of 1857 MB. Nonetheless, the system is stable, and not subject to overuse of resources.

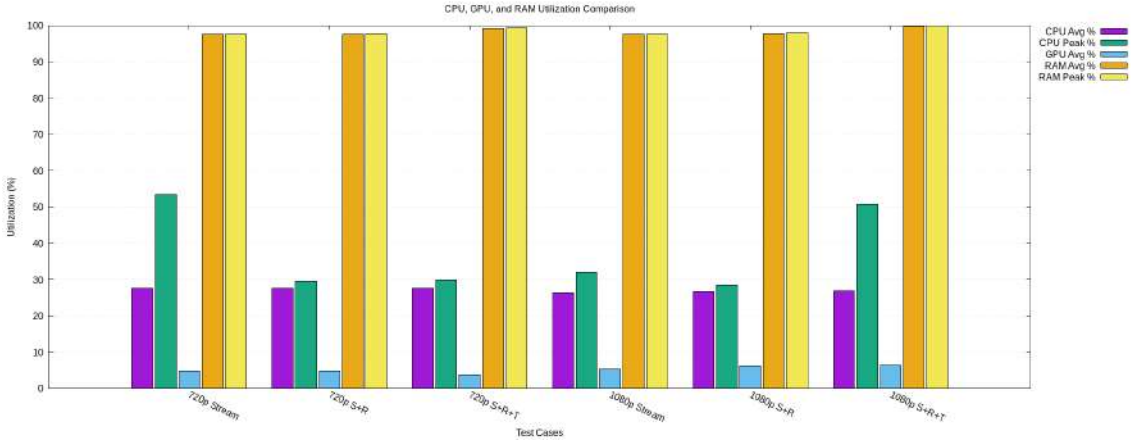


Figure 6.6: Resource Utilization Comparison.

The summary of the resource utilization data is contained in Table 6.1 and the graphical representation of the data in table 6.1 is given in Figure 6.6 in the form of a grouped bar chart. The graph shows the average and peak CPU utilization, average GPU utilization, and normalized average and peak RAM utilization of each test case allowing one to directly compare across the configurations.

The stability of the CPU usage at all the resolutions and operating modes is also clearly depicted in the graphical representation which confirms that CPU is not a limiting factor in the proposed system. The progressive rise between 720p30 and 1080p30 in the usage of the GPU is obvious, which shows the resolution-based character of video encoding loads. Also, the graph highlights the incremental trend of the RAM utilization when recording and telemetry are on, especially at higher resolution. On the whole, the agreement between the table-based data and the trend on the graphs proves the authenticity of the experimental findings and proves that the given architecture effectively manages the resources of the system.

As the analysis of resource utilization shows, the given video streaming, recording, and telemetry system is efficient in all the test configurations. The use of hardware accelerated encoding makes sure that the CPU load is kept low and remains constant, the use of the GPU will increase predictably with the resolution, and the use of memory will increase relatively with the added features. Such findings indicate that it can easily be integrated into real-time embedded systems on NVIDIA Jetson systems, with enough computational and memory capacity to allow future extensions and extended lifetime.

6.2.2 Internal Latency Analysis

Internal latency is the time spent by a video frame to pass through the internal processing stages of a multimedia processing pipeline, beginning at a video capture source and ending at a given processing or output element. Internal latency separates the delay added by the pipeline elements (decoding, colour conversion, encoding, buffering and branching operations) unlike the end-to-end latency at the receiver. Internal latency is also important to measure the pipeline efficiency and to detect the performance bottlenecks, particularly in real-time systems like in the UAV video streaming application.

The internal measurements of latency in this work were measured by the GStreamer latency tracer, which belongs to the built-in tracing framework of GStreamer. Latency tracer captures the latency of each of the buffers in the pipeline that propagates between a source pad and sink pad. It is possible to measure internal delays by turning on the tracer when the pipeline is running, and does not require that the pipeline be changed or any extra timing logic be inserted. This model will provide the correct, non-interfering latency measures that are indicative of the real-life performance.

Test Cases and Measurement Methodology

The latency was measured in two video resolutions, **720p30** and **1080p30**. Three internal pipeline paths of each resolution were considered:

1. **v4l2src - tee**: Latency Measures the latency to the point of branching the pipeline, and it is the minimum processing delay at which the stream can be duplicated.
2. **v4l2src - udpsink**: Measures the internal latency to the streaming output, that is, decoding, conversion, encoding, and RTP packetization.
3. **v4l2src - filesink**: Checks internal latency as far as recording output, encoding and container multiplexing.

The test cases were run with 383 video frames, to have about 12.8 seconds of uninterrupted running time at 30 fpm. Mean latencies were calculated using the tracer samples taken in each of the test cases and statistically stable and similar values were obtained in different configurations of the pipeline.

Results of Latency Analysis

The mean values of the internal latencies were obtained in all the test cases and they are summarized in Table 6.2 and graphically compared in Figure 6.7. In both resolutions, the v4l2src - tee path has the least latency since the path has only necessary preprocessing steps before branching to the pipeline. The mean latency to the tee at 720 p.30 is about 6.6 ms which rises to 7.7 ms at 1080 p.30 because of the

augmented workload on the pixel processing tasks.

The internal latency is much greater with both streaming and recording paths. This is mostly due to the inclusion of hardware-based H.264 encoding and its output processing. The means of the latency of both recording and streaming paths are around 13.3 ms at 720p30 and 20.9 ms at 1080p30. The fact that the values of streaming and recording latency are so closely related means that the encoding part of the latency is mainly the largest part whereas the difference in network transmission and file writing is slightly higher than the overall latency.

Test Case	**Avg. Latency (ms)**
720p30 – `v4l2src` → `tee`	6.58
720p30 – `v4l2src` → `udpsink`	13.25
720p30 – `v4l2src` → `filesink`	13.30
1080p30 – `v4l2src` → `tee`	7.66
1080p30 – `v4l2src` → `udpsink`	20.91
1080p30 – `v4l2src` → `filesink`	20.92

Table 6.2: Average internal pipeline latency for different test cases.

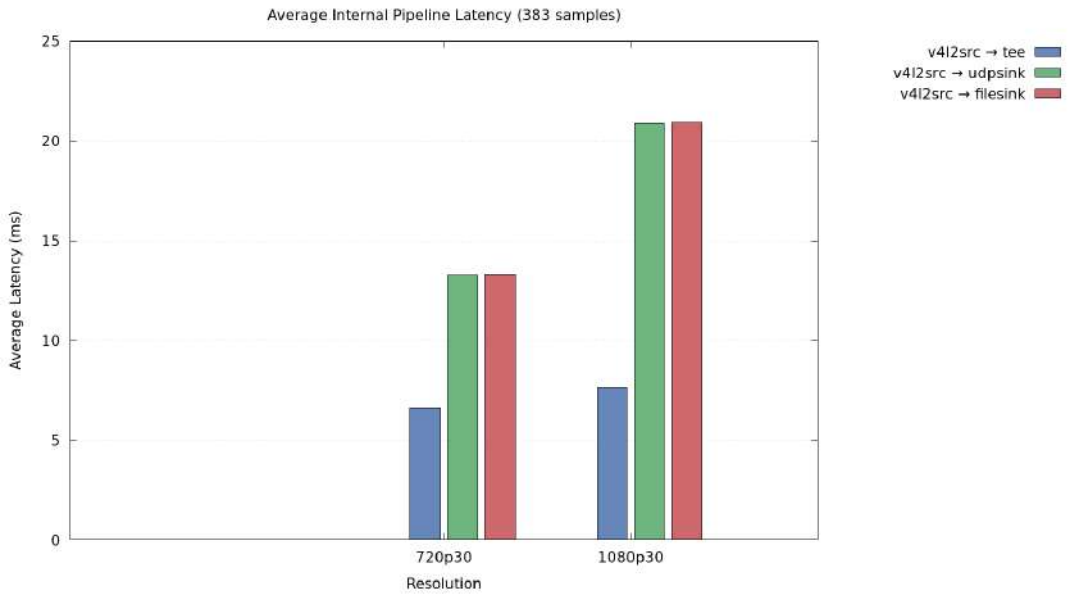


Figure 6.7: Internal latency comparison over test cases.

Figure 6.7 above shows the bar graph that visually indicates how the internal latency is affected by the resolution. The increase in the latency between 720p30

and 1080p30 is the same in all the test cases, which proves that the high spatial resolution directly influences the internal processing time. Although such an increase has occurred the measured latencies are within a range considered acceptable to real-time UAV video streaming applications.

This internal latency analysis shows that major factors that affect pipeline latency are resolution and encoding steps and not the output modality. The latency tracer of GStreamer gives an accurate data on how the internal pipeline operates, allowing a knowledgeable design choice on the optimization of real-time multimedia pipelines in resource-constricted embedded systems.

6.2.3 Frame Continuity and Drop Analysis

The end-to-end frame continuity and frame rate of playback were tested to determine the stability of the suggested UAV video streaming pipeline in the realistic conditions of operating. This analysis has been done with a special receiver-side GStreamer pipeline that ended in fpsdisplaysink. This sink reports runtime information on how many frames have been rendered, how many frames are skipped before being rendered, and the actual playback frame rate, and thus offers a whole end-to-end evaluation of performance of video delivery.

The receiver pipeline was subscribed to the UDP monitoring branch on the port (6000) where the same encoded H.264 video stream is contained as the WebRTC one that drives the browser-based visualization. As the count of frames is incremented when they have been handled by the full receive-side pipeline, which includes RTP reception, jitter buffering, depayloading, decoding, and synchronization, the reported statistics are invariably related to the collective effect of all the upstream processing capabilities, such as camera capture, encoding, packetization, transmission, and scheduling by the receiver. Due to this, the measured values are of realistic end-to-end video delivery properties, as opposed to component-level behavior.

In the course of the measurement, 2500 frames were rendered at the receiver with no dropped frames and this translates to an end-to-end frame drop rate of 0.0%. This finding proves the hypothesis that the suggested streaming pipeline provides a consistent frame continuity, and there is no noticeable loss of frames in the entire delivery chain. No dropped frames means that the system can support the amount of workload configured without overflow of buffers, undue back-pressure in queues or decode side starvation with the conditions tested.

The average playback frame rate at the receiver was found to be 16.65 FPS and the current frame rate at the receiver was about 16.70 FPS towards the end of the measurement period. The playback frame rate was lower than the nominal frame rate set up in the camera input, but does not mean frame loss. Rather, it captures downstream processing and synchronization limitations such as hardware encoder

Metric	Measured Value
Rendered frames	2500
Dropped frames	0
End-to-end drop rate	0.0%
Average playback FPS	16.65

Table 6.3: End-to-end frame continuity and FPS .

scheduling, RTP buffering, decode throughput and sink level clock synchronization (sync=true).

Notably, the fact that the playback frame rate is reduced but maintained constant as well as the fact that frame drop rate is zero suggest that the streaming system is well behaved and is more concerned with continuity and temporal consistency rather than high-performance frame delivery. In the case of UAV situational awareness and remote surveillance, this kind of behavior is usually desirable as compared to increased nominal frame rates with frequent frame drops or visual artifacts. As such, the obtained experimental findings support the hypothesis that the specified video streaming design offers a stable and secure end-to-end video feed that could be utilized in running UAVs live.

6.3 Summary of Evaluation Metrics

This part is a synthesized overview of the outcome of the evaluation performed in both the functional and quantitative analysis discussed in this chapter. The aim of the summary is to present a concise overview of the system performance as per all the major dimensions, such as proper operation, real time behaviour, stability as well as resource efficiency.

Based on the functional assessment, it was possible to note that the system showed the appropriate and stable work of all significant elements. It was successfully possible to stream live video to the browser-based Ground Control Station using the WebRTC, and onboard recording could be initiated and stopped dynamically, without disrupting the live stream. Information (timestamps and telemetry) on geotagging was always included as well in both the streamed and recorded video output. The control interface in a web-based format worked as expected and allowed the control of the streaming and recording process remotely.

The quantitative analysis was also used to verify the actual performance of the system in real-time. According to internal pipeline latency with the GStreamer

latency tracer, steady-state processing delay with pipeline warm-up was very low. The end to end latency had not exceeded acceptable limits in terms of real-time UAV monitoring and streaming stability analysis established that there were no frame-drops and streaming playback showed no obstructing artifacts. Measures of resource usage showed that the system can use the resources efficiently under the computational limits of the NVIDIA Jetson Nano, even when both streaming and recording are done at the same time.

7 Discussion and Future Scope

In this chapter, one can find a detailed discussion of the findings of the implementation and testing of the offered UAV video streaming and recording system. This chapter has the objective of interpreting the results of the experiment and evaluating the effectiveness of the design choices and to correlate the behavior of the observed system with the initial research objectives. Moreover, the possible paths of the future expansion and improvements of the proposed system are described, which also provides the possible prospects of research and development in the future.

7.1 Discussion

The findings achieved in this thesis indicate that a single multimedia processing architecture may be useful in enabling simultaneous real time video streaming, on-demand onboard recording, and geotagging on a resource constrained UAV platform. The implications of the findings of the experiment, the design choices implemented to develop the system, and the context of the experimental findings to the scope of the UAV video systems, are discussed in this section.

An important consequence of this work is that it has become possible to consolidate various video capabilities into one processing pipeline in a coherent way. Prior to the development of the new video recording system, UAVs frequently used separate pipelines or loosely integrated components based on streaming, recording, and telemetry. These designs create a redundancy, augment computation costs and add complexity to video/metadata synchronization. The proposed system, on the contrary, uses a single GStreamer based pipeline where the video capture, preprocessing, encoding, and output distribution is much more closely interrelated. The results of the evaluation prove that such a strategy can bring considerable efficiency without affecting the operational strength.

The choice to make use of the hardware-accelerated video encoding on the NVIDIA Jetson Nano formed part of its core in the successful real-time performance. The qualitative analysis demonstrates that the use of the CPU is not that intensive when the acts of streaming and recording are conducted at the same time, which proves that the implementation of computationally demanding encryption processes is efficiently put on the dedicated hardware units. This is especially necessary in the UAV platforms where the processing resources are strictly constrained and frequently need to be shared with the navigation, control and sensor fusion workloads.

The availability of enough headroom in the computation indicated by the performance of the system on evaluation indicates that the system can be incorporated into larger UAV software stacks without impacting negatively on flight-critical processes.

The discussion has also presented another critical aspect namely latency behavior. Measurement of internal latency in the GStreamer latency tracer showed a huge constant processing delay following initial warm up. These findings suggest that the video processing pipeline is not a bottleneck and most of the end to end delay is caused by encoding, network transmission and decoding at the browsers. The end to end latency stability of the system during continuous operation and state change of the recording mode proves the system design avoids excessive buffering and has predictable timing behavior. Where real-time situational awareness is needed, e.g. in UAV applications, this kind of predictability is frequently a higher priority than absolute latency reduction.

The continuity of the frame and the streaming stability also enhances the strength of the proposed architecture. The lack of dropped frames at the receiver as well the constant playback behavior is a good pointer that the pipeline has a constant throughput and does not experience buffer underflows or overflows. Even though the real-time streaming rate realized in the course of testing was lower than the camera capture rate, it was still fixed and without artifacts. This observation highlights a major research difference between frame rate and frame reliability. Whereas in most instances of UAVs, reliable and steady video transmission is more important than maximum possible frame rate, especially when network or processing resources are limited.

The WebRTC integration through Janus Gateway is an artificial performance or accessibility trade-off. The system ensures a low-latency delivery and it still remains compatible with regular web browsers by sending pre-encoded RTP streams directly into WebRTC sessions without any transcoding. This design option makes client-side requirements easier and can be used to implement Ground Control Station on cross-platform. The testing has verified that this scheme offers consistent video presentation and can withstand client reconnection without interrupting the onboard pipeline which is needed to have operational resilience.

The other area where the discussion indicates the strengths and trade-offs is geo-tagging integration. The direct addition of telemetry data as visual overlay on to video frames makes sure that recorded and live videos are self-contained and can carry contextual data when being transferred or replayed without relying on the UAV system. This is because it does not require the complexity of dealing with parallel metadata streams and also does not cause synchronization problems during post-processing. Although such approach restricts automated metadata extraction, the testing shows that it is a reliable and universally applicable solution that can be used with a variety of operational UAV applications, especially when it comes to

using human operators.

At a system-level, the findings confirm the design philosophy of the design that was driven by simplicity, robustness and efficiency compared to complex features. The average resource overhead that is recorded when recording and the consistency of latency and streaming conditions shows that the system will have a well-balanced trade-off between functionality and performance. This compromise is essential to autonomous UAV operations, where predictability and stability of the system can be more important than more complicated platforms, which are delicate.

Overall, the argument presented above validates that the suggested system will respond to the research questions set in this thesis. The integrated pipeline framework, hardware acceleration, and WebRTC-based streaming and recording of real-time UAV video and geotagging is a viable and scaled solution to real-time UAV video streaming and recording. The experience of this work can be used in the continuous evolution of effective multimedia systems of autonomous aerial platforms and serve as a solid basis when improving the same in the future.

7.2 Future Scope

Despite the fact that the proposed system proves to be quite reliable in its performance and addresses its main goals, it is possible to distinguish a number of directions that can be enhanced in the future. The use of structured metadata formats, e.g., Key-Length-Value (KLV), alongside the current visual overlays is one of the possible directions. This would allow automatic retrieval and processing of the telemetry data and retain the advantages of readable overlays.

This may also be optimized in future by making the pipeline more efficient at giving higher and adjustable frame rates. Modifications to the encoder parameters, camera interfaces, or pipeline buffering policies can enhance the temporal resolution to be used in applications which require a higher frame rate like in a high-speed inspection or tracking task.

Another potential extension is network adaptability. The addition of adaptive bitrate functions or future-generation communication capabilities like 5G might help to increase resilience in changing network conditions and increase working distance. Also, multi-camera-based systems would allow more advanced UAV functions, such as stereo vision, panoramic views or multi-sensor fusion.

Lastly, more sophisticated visualization and analysis capabilities can be added to the web-based Ground Control Station including map-based video overlays, synchronized telemetry dashboards and built-in video replay technology. Such improve-

ments would additionally result in better situational awareness, and help analyzing the mission after the fact more effectively. Collectively, these directions into the future expand on the base set in this thesis and avenue opportunities to more competent and smart UAV video systems.

8 Conclusion

This thesis detailed the design, implementation and testing of a single video processing system to be used with autonomous unmanned aerial vehicles (UAVs), able to handle real time video streaming (simultaneously), on-demand on board recording and embedded geotagging. The overall driving factor of this work was to overcome the shortcomings of current UAV video systems, which tend to use independent and inefficient video processing pipelines to support streaming, video recording, and metadata processing, resulting in a bigger latency, resource utilization, and synchronization cost. In order to pursue the mentioned goals, a modular and at the same time unified multimedia software framework was devised based on the GStreamer framework on an embedded NVIDIA Jetson Nano platform. The described system incorporates video capture, pre-processing, telemetry overlay, Hardware-accelerated H.264 encoding and branching in the output into a single pipeline. Through the use of a tee-based scheme, the video stream to be encoded is effectively transmitted to both a live streaming path as well as a recording path without unnecessary processing. WebRTC enabled through the Janus Gateway made real-time UAV video available to a browser-based Ground Control Station with all the low-latency requirements and platform independence. Functional and quantitative analyses were carried out thoroughly to determine system correctness, performance and efficiency. The findings reveal that the system is stable in supporting continuous live streaming, dynamic start and stop of onboard recording, and geotagging of both streamed and recorded video products. The resource usage metrics also indicated that the system is well within the computational capabilities of the Jetson Nano, even in cases where streaming and recording are both being done.

Within the proposed system telemetry data will be visually superimposed onto the video stream in real time to offer a convenient and dependable way of relating flight data with the image data obtained. By storing mission-relevant parameters in spatial and time, e.g. position, altitude and time stamp, into every video frame, the recorded and streamed video can be self-permeating and can be analyzed without reference to outside telemetry files. This strategy simplifies the review of a post-mission and also provides the ability to provide consistency in interpretation to various video playback environments. Even though the overlay mechanism in place does not offer machine-readable metadata streams, the mechanism is a viable trade-off between implementation simplicity, robustness and usability in operational scenarios, especially in real-time UAV deployments on resource-constrained platforms.

Bibliography

[1] Ahmed, F., Jenihhin, M.: A survey on uav computing platforms: A hardware reliability perspective. Sensors 22(16), 6286 (2022)

[2] Battseren, B., Harradi, R., Kilic, F., Hardt, W.: Automated Power Line Inspection. Technische Universität Chemnitz, Fakultät für Informatik (2020)

[3] Bekmezci, I., Sahingoz, O.K., Temel, Ş.: Flying ad-hoc networks (fanets): A survey. Ad Hoc Networks 11(3), 1254–1270 (2013)

[4] Bentaleb, A., Lim, M., Akcay, M.N., Begen, A.C., Hammoudi, S., Zimmermann, R.: Toward one-second latency: Evolution of live media streaming. IEEE Communications Surveys & Tutorials (2025)

[5] Bentaleb, A., Lim, M., Akcay, M.N., Begen, A.C., Hammoudi, S., Zimmermann, R.: Toward one-second latency: Evolution of live media streaming. IEEE Communications Surveys & Tutorials (2025)

[6] Bhatt, N., Thakkar, A.: An efficient approach for low latency processing in stream data. PeerJ Computer Science 7, e426 (2021)

[7] Briley, A.A., Afghah, F.: Hardware acceleration for real-time wildfire detection onboard drone networks. In: IEEE INFOCOM 2024-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). pp. 01–06. IEEE (2024)

[8] Cao, H., Xu, J., Li, D., Yang, Z., Liu, Y.: Eventboost: Event-based acceleration platform for real-time drone localization and tracking. In: IEEE INFOCOM 2024-IEEE Conference on Computer Communications. pp. 1851–1859. IEEE (2024)

[9] Cass, S.: Nvidia makes it easy to embed ai: The jetson nano packs a lot of machine-learning power into diy projects-[hands on]. IEEE Spectrum 57(7), 14–16 (2020)

[10] De Cicco, L., Mascolo, S., Palmisano, V.: Feedback control for adaptive live video streaming. In: Proceedings of the second annual ACM conference on Multimedia systems. pp. 145–156 (2011)

[11] Elmanaa, I., Sabri, M.A., Abouch, Y., Aarab, A.: Efficient roundabout supervision: real-time vehicle detection and tracking on nvidia jetson nano. Applied sciences 13(13), 7416 (2023)

[12] Grüner, M., Licciardello, M., Singla, A.: Reconstructing proprietary video streaming algorithms. In: 2020 USENIX Annual Technical Conference (USENIX ATC 20) (2020)

[13] GStreamer Project: GStreamer Application Development Manual. `https://gstreamer.freedesktop.org/documentation/application-development/introduction/gstreamer.html` (2024), accessed: 2026-01-10

[14] Gupta, C., Patra, T.K.: Integrating klv metadata with uav motion imagery and geo-tagging for enhanced gis playback. In: Proceedings of the 3rd International Conference for Advancement in Technology (ICONAT). pp. 1–5. IEEE (2024)

[15] Herrero, R.: Reliable transmission of stream transported media in wireless real time communications. Wireless Networks 25(8), 4727–4736 (2019)

[16] Ijaz, H., Ahmad, R., Ahmed, R., Ahmed, W., Kai, Y., Jun, W.: A uav-assisted edge framework for real-time disaster management. IEEE Transactions on Geoscience and Remote Sensing 61, 1–13 (2023)

[17] Istomin, T., Leoni, E., Molteni, D., Murphy, A.L., Picco, G.P., Griva, M.: Janus: Dual-radio accurate and energy-efficient proximity detection. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies 5(4), 1–33 (2021)

[18] Jacob, J., Elayidom, M.S., Devassia, V.P.: An innovative approach for aerial video surveillance using video content analysis and indexing. In: International Conference on Image Processing and Capsule Networks. pp. 574–583. Springer (2020)

[19] Jin, J., Ma, J., Liu, L., Lu, L., Wu, G., Huang, D., Qin, N.: Design of uav video and control signal real-time transmission system based on 5g network. In: 2021 IEEE 16th Conference on Industrial Electronics and Applications (ICIEA). pp. 533–537. IEEE (2021)

[20] Kacianka, S., Hellwagner, H.: Adaptive video streaming for uav networks. In: Proceedings of the 7th ACM international workshop on mobile video. pp. 25–30 (2015)

[21] Kakkavas, G., Diamanti, M., Karyotis, V., Nyarko, K.N., Gabriel, M., Zafeiropoulos, A., Papavassiliou, S., Moessner, K.: 5g perspective of connected autonomous vehicles: Current landscape and challenges toward 6g. IEEE Wireless Communications 31(4), 299–306 (2024)

[22] Kalan, R., Dulger, I.: A survey on qoe management schemes for http adaptive video streaming: challenges, solutions, and opportunities. IEEE Access (2024)

[23] Kilic, F., Hassan, M., Hardt, W.: Prototype for multi-uav monitoring–control system using webrtc. Drones 8(10), 551 (2024)

[24] Kim, S.G., Lee, E., Hong, I.P., Yook, J.G.: Review of intentional electromagnetic interference on uav sensor modules and experimental study. Sensors 22(6), 2384 (2022)

[25] Kim, W.J., Jang, H., Choi, G.B., Hwang, I.S., Youn, C.H.: A webrtc based live streaming service platform with dynamic resource provisioning in cloud. In: 2016 IEEE Region 10 Conference (TENCON). pp. 2424–2427. IEEE (2016)

[26] Kumar, T., Sharma, P., Tanwar, J., Alsghier, H., Bhushan, S., Alhumyani, H., Sharma, V., Alutaibi, A.I.: Cloud-based video streaming services: Trends, challenges, and opportunities. CAAI Transactions on Intelligence Technology 9(2), 265–285 (2024)

[27] Leow, R.: Webrtc-based video quality of experience evaluation of the janus streaming plugin: Integrating video door systems and webrtc-supported browsers (2018)

[28] Lin, Y.C., Wu, S.C.: An accelerated h. 264/avc encoder on graphic processing unit for uav videos. In: International Symposium Computational Modeling of Objects Represented in Images. pp. 251–258. Springer (2016)

[29] Liu, Y., Du, B., Wang, S., Yang, H., Wang, X.: Design and implementation of performance testing utility for rtsp streaming media server. In: 2010 First International Conference on Pervasive Computing, Signal Processing and Applications. pp. 193–196. IEEE (2010)

[30] Lu, Y., To, H., Alfarrarjeh, A., Kim, S.H., Yin, Y., Zimmermann, R., Shahabi, C.: Geougv: User-generated mobile video dataset with fine granularity spatial metadata. In: Proceedings of the 7th International Conference on Multimedia Systems. pp. 1–6 (2016)

[31] Luo, J., Joshi, D., Yu, J., Gallagher, A.: Geotagging in multimedia and computer vision—a survey. Multimedia Tools and Applications 51(1), 187–211 (2011)

[32] Marques, O.: Practical image and video processing using MATLAB. John Wiley & Sons (2011)

[33] Mittal, S.: A survey of techniques for improving energy efficiency in embedded computing systems. International Journal of Computer Aided Engineering and Technology 6(4), 440–459 (2014)

[34] Mogaka, O.M., Zewail, R., Inoue, K., Sayed, M.S.: Tinyemergencynet: a hardware-friendly ultra-lightweight deep learning model for aerial scene image classification. Journal of Real-Time Image Processing 21(2), 51 (2024)

[35] Morgenthal, G., Hallermann, N.: Quality assessment of unmanned aerial vehicle (uav) based visual inspection of structures. Advances in Structural Engineering 17(3), 289–302 (2014)

[36] Nimmi, S., Saranya, V., Gandhiraj, R., et al.: Real-time video streaming using gstreamer in gnu radio platform. In: 2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE). pp. 1–6. IEEE (2014)

[37] Nomani, T., Mohsin, M., Pervaiz, Z., Shafique, M.: xuavs: Towards efficient approximate computing for uavs—low power approximate adders with single lut delay for fpga-based aerial imaging optimization. IEEE Access 8, 102982–102996 (2020)

[38] NVIDIA Corporation: NvMedia Image Processing Pipeline (IPP) Concept. `https://docs.nvidia.com/drive/drive_os_5.1.6.1L/nvvib_docs/index.html#page/DRIVE_OS_Linux_SDK_Development_Guide/NvMedia/nvmedia_concept_ipp.html` (2021), accessed: 2026-01-10

[39] O'Connor, N.E., Duffy, T., Ferguson, P., Gurrin, C., Lee, H., Sadlier, D.A., Zhang, K.: A content-based retrieval system for uav-like video and associated metadata. In: Airborne Intelligence, Surveillance, Reconnaissance (ISR) Systems and Applications V. vol. 6946, pp. 126–135. SPIE (2008)

[40] Paolucci, F., Sgambelluri, A., Cugini, F., Castoldi, P.: Network telemetry streaming services in sdn-based disaggregated optical networks. Journal of Lightwave Technology 36(15), 3142–3149 (2018)

[41] Patel, S., Mohan, A., Kaur, B., Mathur, A., Pateriya, B.: Geo-tagged video visualization using open source web-gis techniques for road condition monitoring. Journal of Geomatics 19(1), 15–22 (2025)

[42] Peltotalo, J., Harju, J., Väätämöinen, L., Bouazizi, I., Curcio, I.D.: Rtsp-based mobile peer-to-peer streaming system. International Journal of Digital Multimedia Broadcasting 2010(1), 470813 (2010)

[43] Peroni, L., Gorinsky, S.: An end-to-end pipeline perspective on video streaming in best-effort networks: a survey and tutorial. ACM Computing Surveys 57(12), 1–47 (2025)

[44] Philp, A., Bradley, B., Stach, J., Rodriguez, T.: Real-time application of digital watermarking to embed tactical metadata into full motion video captured from unmanned aerial systems. In: Media Forensics and Security. vol. 7254, pp. 321–333. SPIE (2009)

[45] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y., et al.: Ros: an open-source robot operating system. In: ICRA workshop on open source software. vol. 3, p. 5. Kobe (2009)

[46] Roglia, E.: Geographical map annotation with social metadata in a surveillance environment. International Journal of Computer Applications (2010)

[47] Sacoto-Martins, R., Madeira, J., Matos-Carvalho, J.P., Azevedo, F., Campos, L.M.: Multi-purpose low latency streaming using unmanned aerial vehicles. In: 2020 12th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP). pp. 1–6. IEEE (2020)

[48] Safin, R., Garipova, E., Lavrenov, R., Li, H., Svinin, M., Magid, E.: Hardware and software video encoding comparison. In: 2020 59th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE). pp. 924–929. IEEE (2020)

[49] Salehe, M., Hu, Z., Mortazavi, S.H., Mohomed, I., Capes, T.: Vi@deopipe: Building video stream processing pipelines at the edge. In: Proceedings of the 20th international middleware conference industrial track. pp. 43–49 (2019)

[50] Sharma, M.K., Farhat, I., Liu, C.F., Sehad, N., Hamidouche, W., Debbah, M.: Real-time immersive aerial video streaming: A comprehensive survey, benchmarking, and open challenges. IEEE Open Journal of the Communications Society (2024)

[51] Stephan, M., Battseren, B., Hardt, W.: Object Localization in 3D Space for UAV Flight Using a Monocular Camera. Ph.D. thesis, Technische Universität Chemnitz, Fakultät für Informatik (2021)

[52] Stephan, M., Battseren, B., Tudevdagva, U.: Autonomous unmanned aerial vehicle development: Mavlink abstraction layer. In: International Symposium on Computer Science, Computer Engineering and Educational Technology (ISCSET-2020), Lauta Germany. pp. 45–49 (2020)

[53] Stockhammer, T.: Dynamic adaptive streaming over http– standards and design principles. In: Proceedings of the second annual ACM conference on Multimedia systems. pp. 133–144 (2011)

[54] Sulema, Y., Amram, N., Aleshchenko, O., Sivak, O.: Quality of experience estimation for webrtc-based video streaming. In: European Wireless 2018; 24th European Wireless Conference. pp. 1–6. VDE (2018)

[55] Valladares, S., Toscano, M., Tufiño, R., Morillo, P., Vallejo-Huanga, D.: Performance evaluation of the nvidia jetson nano through a real-time machine learning application. In: International Conference on Intelligent Human Systems Integration. pp. 343–349. Springer (2021)

[56] Viola, R., Martin, A., Zorrilla, M., Montalban, J., Angueira, P., Muntean, G.M.: A survey on virtual network functions for media streaming: Solutions and future challenges. ACM Computing Surveys 55(11), 1–37 (2023)

[57] Wang, B., Peng, X., Jiang, M., Liu, D.: Real-time fault detection for uav based on model acceleration engine. IEEE Transactions on Instrumentation and Measurement 69(12), 9505–9516 (2020)

[58] Wen, Z., Yang, R., Qian, B., Xuan, Y., Lu, L., Wang, Z., Peng, H., Xu, J., Zomaya, A.Y., Ranjan, R.: Janus: Latency-aware traffic scheduling for iot data streaming in edge environments. IEEE Transactions on Services Computing 16(6), 4302–4316 (2023)

[59] Wengle, E., Erstorp, E.S., Lidström, V., Varagnolo, D., Dong, H.: Experimental assessment of a janus-based consensus protocol. Computer Networks 244, 110345 (2024)

[60] Yin, X., Jindal, A., Sekar, V., Sinopoli, B.: A control-theoretic approach for dynamic adaptive video streaming over http. In: Proceedings of the 2015 ACM conference on special interest group on data communication. pp. 325–338 (2015)

[61] Zhu, Y., Liu, H., Guo, Y., Zeng, W.: Network assisted media streaming in multi-hop wireless networks. In: 2011 Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN). pp. 1–7. IEEE (2011)

References: Professorship of Computer Engineering

[23] Kilic, F., Hassan, M., Hardt, W.: Prototype for multi-UAV monitoring–control system using WebRTC. *Drones* 8(10), 551 (2024).

[52] Stephan, M., Battseren, B., Tudevdagva, U.: Autonomous unmanned aerial vehicle development: MAVLink abstraction layer. In: *ISCSET-2020*, pp. 45–49 (2020).

[51] Stephan, M., Battseren, B., Hardt, W.: Object Localization in 3D Space for UAV Flight Using a Monocular Camera. Ph.D. thesis, Technische Universität Chemnitz (2021).

[2] Battseren, B., Harradi, R., Kilic, F., Hardt, W.: Automated Power Line Inspection. Technische Universität Chemnitz, Fakultät für Informatik (2020).

[21] Kakkavas, G., Diamanti, M., Karyotis, V., Nyarko, K. N., Gabriel, M., Zafeiropoulos, A., Papavassiliou, S., Moessner, K.: 5G perspective of connected autonomous vehicles: Current landscape and challenges toward 6G. *IEEE Wireless Communications* 31(4), 299–306 (2024).

TECHNISCHE UNIVERSITÄT CHEMNITZ

Zentrales Prüfungsamt

Selbstständigkeitserklärung

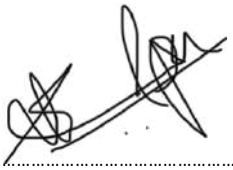
Name: Geetha Harshakumar Vorname: Silpa geb. am: 27.09.1993 Matr.-Nr.: 714848	**Bitte beachten:** 1. Bitte binden Sie dieses Blatt am Ende Ihrer Arbeit ein.

Selbstständigkeitserklärung*

Ich erkläre gegenüber der Technischen Universität Chemnitz, dass ich die vorliegende Masterarbeit selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe.

Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch nicht als Prüfungsleistung eingereicht und ist auch noch nicht veröffentlicht.

Datum: 05.02.2026 Unterschrift: 

* Statement of Authorship

I hereby certify to the Technische Universität Chemnitz that this thesis is all my own work and uses no external material other than that acknowledged in the text.

This work contains no plagiarism and all sentences or passages directly quoted from other people's work or including content derived from such work have been specifically credited to the authors and sources.

This paper has neither been submitted in the same or a similar form to any other examiner nor for the award of any other degree, nor has it previously been published.

This report - except logo Chemnitz University of Technology - is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made. The images or other third party material in this report are included in the report's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the report's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this license, visit http://creativecommons.org/licenses/by/4.0/.

Chemnitzer Informatik-Berichte

In der Reihe der Chemnitzer Informatik-Berichte sind folgende Berichte erschienen:

CSR-25-01 Md. Ali Awlad, Hasan Saadi Jaber Aljzaere, Wolfram Hardt, AUTOSAR Software Component for Atomic Straight Driving Patterns, März 2025, Chemnitz

CSR-25-02 Billava Vasantha Monisha, Hasan Saadi Jaber Aljzaere, Wolfram Hardt, Automotive Software Component for QT Based Car Status Visualization, März 2025, Chemnitz

CSR-25-03 Zahra Khadivi, Batbayar Battseren, Wolfram Hardt, Acoustic-Based MAV Propeller Inspection, Mai 2025, Chemnitz

CSR-25-04 Tripti Kumari Shukla, Ummay Ubaida Shegupta, Wolfram Hardt, Time Management Tool Development to Support Self-regulated Learning, August 2025, Chemnitz

CSR-25-05 Ambu Babu, Ummay Ubaida Shegupta, Wolfram Hardt, Development of a Retrieval Model based Backend of a Tutoring Agent, August 2025, Chemnitz

CSR-25-06 Shahid Ismail, Ummay Ubaida Shegupta, Wolfram Hardt, Development of a Generative Model based Backend of Tutoring Agent, August 2025, Chemnitz

CSR-25-07 Chaitanya Sravanthi Akula, Ummay Ubaida Shegupta, Wolfram Hardt, Integration of Learning Analytics into the ARC-Tutoring Workbench, August 2025, Chemnitz

CSR-25-08 Jörn Roth, Reda Harradi, Wolfram Hardt, Implementation of a Path Planning Algorithm for UAV Navigation, Dezember 2025, Chemnitz

CSR-25-09 Alhassan Khalil, Reda Harradi, Stephan Rupf, Wolfram Hardt, Development of an Automation Framework for 1D Measurement, Dezember 2025, Chemnitz

CSR-26-01 Vismay Gunda, Shadi Saleh, Wolfram Hardt, Cloud-Based AI Solutions for Ensuring Data Quality in Predictive Models, Februar 2026, Chemnitz

CSR-26-02 Sami Mansoor Alavi, Shadi Saleh, Wolfram Hardt, Continuous Integration for Cloud-Based Swarm Farming Applications, Februar 2026, Chemnitz

Chemnitzer Informatik-Berichte

CSR-26-03 Sarah Onyinyechi Obasi, Shadi Saleh, Wolfram Hardt, Cloud-Based AI for Data Completeness Analysis and Improvement in Predictive Modeling, März 2026, Chemnitz

CSR-26-04 Atul Chandra Nath, Reda Harradi, Wolfram Hardt, GPS-based UAV Precision Landing, April 2026, Chemnitz

CSR-26-05 Josey Mol Sibi, Batbayar Battseren, Wolfram Hardt, Real Time Multi Stream Video Transmission in Autonomous UAV, Mai 2026, Chemnitz

CSR-26-06 Silpa Geetha Harshakumar, Batbayar Battseren, Wolfram Hardt, Simultaneous Video Streaming and Recording with Geotagging in Autonomous UAV, Mai 2026, Chemnitz

Chemnitzer Informatik-Berichte

ISSN 0947-5125

Herausgeber: Fakultät für Informatik, TU Chemnitz
Straße der Nationen 62, D-09111 Chemnitz