TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Aleksei Lutkov 243440IAIB

Integration and Optimization of Computer Vision Models for Fish Species, Size, and Behavior Classification in the Smart Fish Counter Application

Bachelor's Thesis

Supervisor: Jeffrey A. Tuhtan
Tenured Associate Professor Dr.-Eng.

Tallinn 2025

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Aleksei Lutkov 243440IAIB

KALA LIIGI, SUURUSE JA KÄITUMISE KLASSIFIKATSIOONI ARVUTINÄGEMISMUDELITE INTEGREERIMINE JA OPTIMEERIMINE SMART FISH COUNTER RAKENDUSES

Bakalaureusetöö

Juhendaja: Jeffrey A. Tuhtan
Kaasprofessor tenuuris Dr.-Eng.

Tallinn 2025

Author’s Declaration of Originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Aleksei Lutkov

12.01.2025

Abstract

Climate change and anthropogenic impacts are leading to a rapid decline in freshwater biodiversity worldwide, which has decreased by more than 80% since 1970. Technologies are therefore urgently needed to automate freshwater biodiversity monitoring, and the TalTech Smart Fish Counter Application has been developed to leverage the most recent advances in computer vision methods for fish detection, size and species classification, and migration behavior analysis.

In this bachelor's thesis, the author developed the Smart Fish Counter Application further. The objectives of this work include integrating three new ML (Machine Learning) modules for fish species, size, and behavior classification, testing and validating the integrated modules, and optimizing the application's computational efficiency. The author addresses the challenges of integrating these new modules into the application and improves the performance of the individual modules, leading to a substantial reduction in the video processing times.

The results of this work include a fully functional, new Smart Fish Counter Application with integrated, tested, and validated algorithms for fish species, size, and behavior classification and includes an overview of the optimization, which reduced the average processing time from 216 seconds to 27 seconds per video.

The thesis is in English and contains 35 pages of text, 6 chapters, 9 figures, 2 tables.

Annotatsioon

Kala liigi, suuruse ja käitumise klassifikatsiooni arvutinägemismudelite integreerimine ja optimeerimine Smart Fish Counter rakenduses

Kliimamuutus ja antropogeensed tegurid kahandavad kiiresti magevee bioloogilist mitmekesisust kogu maailmas, mis 1970. aastast on vähenenud enam kui 80% võrra. Selle kahanemise tõttu on magevee bioloogilise mitmekesisuse seire automatiseerimiseks hädasti vaja tehnoloogiaid nagu TalTech Smart Fish Counter Application. See rakendus oli välja töötanud arvutinägemise meetodite uusimate edusammude ärakasutamiseks ja kujutab endast avatud lähtekoodiga seiresüsteemi keskkonnatingimuste hindamiseks, kalade loendamiseks, nende rändekäitumise uurimiseks ning nende suuruse ja liigi tuvastamiseks.

See bakalaureusetöö on pühendatud Smart Fish Counter Application edasiarendamisele. Selle eesmärgid hõlmavad kolme masinõppe mooduli integreerimist kala rändekäitumise, liigi ja suuruse klassifitseerimiseks, integreeritud moodulite testimist ja valideerimist ning rakenduse arvutusliku efektiivsuse optimeerimist. Lõputöö vältel autor käsitleb uute moodulite integreerimisega seotud väljakutseid, nagu tihe seos moodulite vahel ja videotöötlusaegade optimeerimine.

Selle töö tulemused hõlmavad esiteks, täielikult funktsionaalset uut rakenduse versiooni integreeritud, testitud ja valideeritud algoritmidega kala rändekäitumise, liigi ja suuruse klassifitseerimiseks ning teiseks, ülevaadet teostatud optimeerimistest, mille käigus suutis autor vähendada keskmist videotöötlusaega 216 sekundilt 27 sekundile.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 35 leheküljel, 6 peatükki, 9 joonist, 2 tabelit.

List of Abbreviations and Terms

AI	Artificial Intelligence
BfG	*Bundesanstalt für Gewässerkunde*. German Federal Institute of Hydrology
BRAIN	Biotactic Research Artificial Intelligence Network
CPU	Central Processing Unit
DB	Database
FC	Fish Counter
FNF	Fish/No-Fish
FPS	Frames Per Second
GIL	Global Interpreter Lock
GPU	Graphics Processing Unit
HD	High-Definition
HPC	High-Performance Computing
IO	Input/Output
ML	Machine Learning
OS	Operating System
QA	Quality Assurance
SPA	Single-Page Application
SSR	Server-Side Rendering
UI	User Interface
VM	Virtual Machine
YOLO	You Only Look Once

Table of Contents

List of Figures

List of Tables

1 Introduction

The Smart Fish Counter is an open source full-stack web application developed by TalTech's Center for Environmental Intelligence and Sensing for the BfG (*Bundesanstalt für Gewässerkunde*, German Federal Institute of Hydrology). It is the most advanced ML system for the detection of freshwater fish in Europe and provides a versatile tool for the identification of individual fish species and the study of their migration behavior, replacing most of the expensive and time-consuming tasks currently performed by trained biologists to evaluate fish migration [1].

The application includes five core computer vision modules. The first two modules were integrated before this thesis work, and the remaining three are in development and have not yet been integrated into the application. Module 1 classifies videos according to six commonly occurring environmental conditions, sorting the videos for further processing. Module 2 labels videos with fish and videos without fish. Module 3 detects the species and family of fish. Module 4 determines the size class of the fish. Module 5 classifies eight different migration behaviors of individual fish during their passage through the counter.

The main contributions of this work relate to integrating the computer vision algorithms for modules 3, 4, and 5 and optimizing the video processing time. Due to the technical challenges associated with integrating the remaining modules, they have not yet been integrated into the application. The trained personnel manually define the classifications for which those modules are responsible. The major challenges addressed in this work are:

- **Interdependence between the modules.** The three remaining modules are not completely independent of each other, which poses a challenge in their integration. For example, the species module relies on the previously determined size of the fish to correct species estimations based on biological studies that determine maximum and minimum species sizes. The fish behavior module also uses species information. For example, some small-bodied fish tend to migrate in groups of five or more fish at a time. In contrast, other species, such as European eel, prefer to migrate alone.

- **Video processing time.** The full validation dataset consists of more than 400,000 human-annotated videos completed by eight human raters over 2.5 years, with more than 40 terabytes of video material. The current training dataset for all modules consists of several hundred gigabytes, which results in large network weights. Although large models tend to perform well because of their inherent flexibility, the speed of processing videos in the application is reduced. Therefore, the performance of the application must be optimized after the remaining modules have been integrated.

The main objectives of this bachelor's thesis are three-fold:

- Integration of computer vision ML algorithms for the automated detection of fish species, size class, and individual fish behavioral evaluation;
- Testing and validation of the integrated algorithms;
- Optimization of the application's ML algorithms to improve computational efficiency and reduce processing times.

The primary result of the work is a new version of the web application with integrated, tested, validated, and optimized computer vision algorithms for the automated detection of fish species, size class, and individual fish behavior classification. The secondary result is the comparative analysis of optimization methods tested by the author to improve the computational efficiency of the application's computer vision algorithms.

A literature review is used in this work to give an overview of the background, examine existing analogous solutions, and clearly establish the significance, novelty, and contributions of the thesis topic. To manage the project related to this bachelor's thesis and achieve set objectives and results, a simplified version of Scrum methodology [2] is used, according to which increments of work are delivered in short cycles called sprints. Validation of the modules is based on a documented QA (Quality Assurance) process for each module separately. The BfG also provided direct feedback throughout this thesis work on the model integration and processing times, as they have a stand-alone web application installation on their own HPC (High-Performance Computing) center in Koblenz, Germany.

2 Background

In this chapter, background information is provided explaining the need and purpose of automated fish biodiversity monitoring systems. An overview and comparison of the existing river fish counter systems are also given in this chapter, explaining the need for the development of the Smart Fish Counter Application.

2.1 Motivation

Freshwater ecosystems, home to approximately a third of all vertebrate species, are rapidly declining, with global wetland destruction occurring three times faster than forest loss [1]. In addition, the combined effects of climate and human changes are causing freshwater vertebrate populations to decline more than twice as fast as terrestrial or marine populations.

Taking into account this trajectory, measures must be taken to bring the situation under control, and monitoring of freshwater biodiversity could play a crucial role in slowing the rate of decline in the freshwater vertebrate population. Automated freshwater biodiversity monitoring systems are therefore urgently needed to examine, predict, and support sustainable freshwater fisheries, as well as assess the effectiveness of conservation actions.

2.2 River Fish Counters

River fish counter systems are installed to support freshwater biodiversity monitoring by studying fish migration during their passage through the counter. They are usually strategically placed at fish passes and other natural or man-made narrow points, such as dams, weirs, and bypasses, where fish tend to migrate.

Multiple technologies are used for river fish counter systems and could be divided into the following types: hydroacoustic counters, resistivity counters, optical beam counters, and video counters [3]. In this work, the focus is maintained only on the video fish counters because the Smart Fish Counter represents this type of counter.

Several different video river fish counter systems, each with unique features, are available to increase the ease and rate of video processing. The following is an overview of the five most dominant systems currently available, including their descriptions and characteristics.

2.2.1 VAKI Riverwatcher

VAKI Riverwatcher fish counter system is developed by an Icelandic company, Vaki Aquaculture Systems Ltd., and uses infrared scanning technology and high-resolution cameras to monitor fish migration during their passage through the counter [4], [5]. The Riverwatcher automatically counts the fish, measures their size and saves fish silhouette images or videos, which can be later used by users to identify fish species.

The system includes a scanner on a river and a weatherproof control unit on the river bank. The scanner data are sent to the control unit for storage and can be analyzed by users with the proprietary software WinAri, also developed by Vaki Aquaculture Systems Ltd.

2.2.2 Simsonar FC

Simsonar FC (Fish Counter) is a product of the Finnish company Simsonar Oy [6]. The system utilizes a 3D stereo camera to capture HD (High-Definition) fish videos as the fish passes through the counter. The system also uses neural networks with cloud-based services to analyze fish movement and behavior and classify their species and size.

The system features cloud-based services for species identification, allowing Simsonar FC to identify the twelve most common species of Nordic fish and recognize adipose salmon and trout fins, indicating that the fish was not cultivated.

2.2.3 TiVA FC

TiVA FC is a fish counting and monitoring system developed by the Swedish company TiVA AB [7], [8]. The system is coupled with a dedicated cloud-based web platform for real-time data processing using AI (Artificial Intelligence) algorithms. It can track and analyze fish movement, identify species and size, and detect adipose fin and fungal growth.

The TiVA FC has an HD video camera for capturing passing fish video data and a 3D camera for precise fish size measurement. The initial data processing is performed by embedded hardware, which also uploads the data to the web platform for further processing.

2.2.4 BRAVO G3

BRAVO G3 is a fish monitoring system developed by a Canadian company, Biotactic Inc. [9]. It employs HD cameras with built-in infrared illumination to improve video capture at night and a self-cleaning lens system that eliminates the need for underwater maintenance.

It is possible to connect BRAVO G3 to the Internet, allowing live uploading of video data to central BRAVO computers or other desired locations. Central BRAVO computers process video data with Biotactic's proprietary AI software, BRAIN (Biotactic Research Artificial Intelligence Network). BRAIN allows counting fish and identifying their movement direction, but species identification and size estimation must be performed manually.

2.2.5 I AM HYDROCam

I AM HYDROCam, developed by I AM HYDRO GmbH, is an underwater camera system for aquaculture, fisheries management, and ecological research, used to monitor fish migration [10]. It is a dual-sensor camera system that features separate day and night sensors that allow the system to operate under varying lighting conditions [11].

The I AM HYDROCam manufacturer has partnered with TalTech's Center for Environmental Intelligence and Sensing. It is the first camera system to use the modules integrated into the Smart Fish Counter Application on the embedded hardware on the camera. Currently, it processes videos using an AI module to sort videos with and without fish. In the future, it will also use the modules to classify fish species, size, and migration behavior.

2.2.6 Counters Comparison

Each of the commercial video river fish counter systems discussed has unique features. Table 1 shows general information about each of these systems.

Table 1. Commercial video river fish counter systems comparison.

	VAKI Riverwatcher	**Simsonar FC**	**TiVA FC**	**BRAVO G3**	**I AM HYDROCam**
Tracking	Yes	Yes	Yes	Yes	Yes
Behavior	Yes	Yes	Yes	Yes	Yes
Size	Yes	Yes	Yes	No	Yes
Species	No	Yes	Yes	No	Yes
Software location	Local installation	Local installation	Cloud-based	Cloud-based	Cloud-based
Software type	Closed-source	Closed-source	Closed-source	Closed-source	Open-source
Environmental conditions	No	No	No	No	Yes
Independent manual validation	No	No	No	No	Yes

Although existing systems are sophisticated and advanced, they still lack the features needed for their efficient and convenient use. For example, not all of the systems discussed can identify the size and species of fish. Similarly, the software for some of the systems has to be installed locally, which is inconvenient compared to using cloud-based solutions.

Even more importantly, only the software coupled with the I AM HYDROCam system is open source, which greatly reduces the development potential because interested parties cannot make contributions. Noteworthy is also that, other than the I AM HYDROCam, existing systems do not contain environmental conditions and do not allow the user to validate the system's performance independently of the ML models they apply.

Due to these reasons, there is a need for a more transparent fish monitoring tool that is released open source and includes environmental conditions during data processing, and allows the user to independently validate the system's performance.

3 Methods

This chapter describes the Smart Fish Counter Application state before the work on this bachelor's thesis and outlines the requirements and technical features of the work.

3.1 Technology Overview

The Smart Fish Counter Application is a complex, full-stack system developed according to modern software development practices. The application's front-end represents a SPA (Single-Page Application) written in the React Library and pre-rendered on the server side, where the Razzle build tool, Node.js runtime, and Koa.js framework handle the SSR (Server-Side Rendering). The back-end of the application represents a complex system combining technologies such as Django, Celery, Redis, and PostgreSQL.

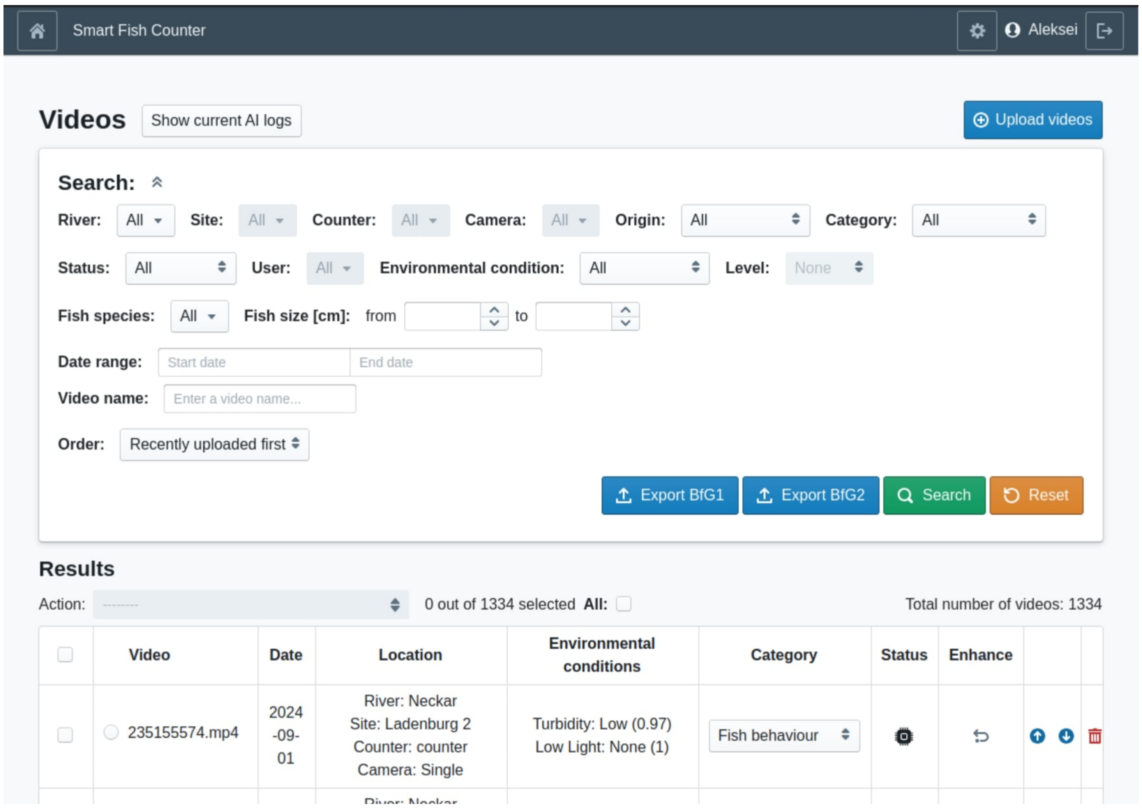
In addition to the technologies used in the application's front-end and back-end, the application also uses Docker. This open source tool enables the building, running, and managing of containers that combine the source code with the libraries and dependencies of the OS (Operating System) [12]. This allows the code to run in any environment and provides isolation and scalability of applications.

Furthermore, the application also uses Ansible, an open source automation software written in Python that enables infrastructure as code and features software provisioning, configuration management, and application deployment [13]. In the Smart Fish Counter Application context, it is used to configure the server host and deploy the application to it.

3.2 Initial Application State

The Smart Fish Counter Application is an open science development project that started in early 2021. Its goal is to create versatile open source ML software with a web application UI (User Interface) for use with freshwater fish counters that automatically detects environmental conditions in processed videos and allows the user to validate the system's performance independently.

Before this bachelor's thesis, a team of developers had already worked on the application and implemented its general functionality. Since the application is a SPA, all its primary functionality is available from the main application view, as illustrated in Figure 1.



River: All · Site: All · Counter: All · Camera: All · Origin: All · Category: All

Status: All · User: All · Environmental condition: All · Level: None

Fish species: All · Fish size [cm]: from … to …

Date range: Start date · End date

Video name: Enter a video name...

Order: Recently uploaded first

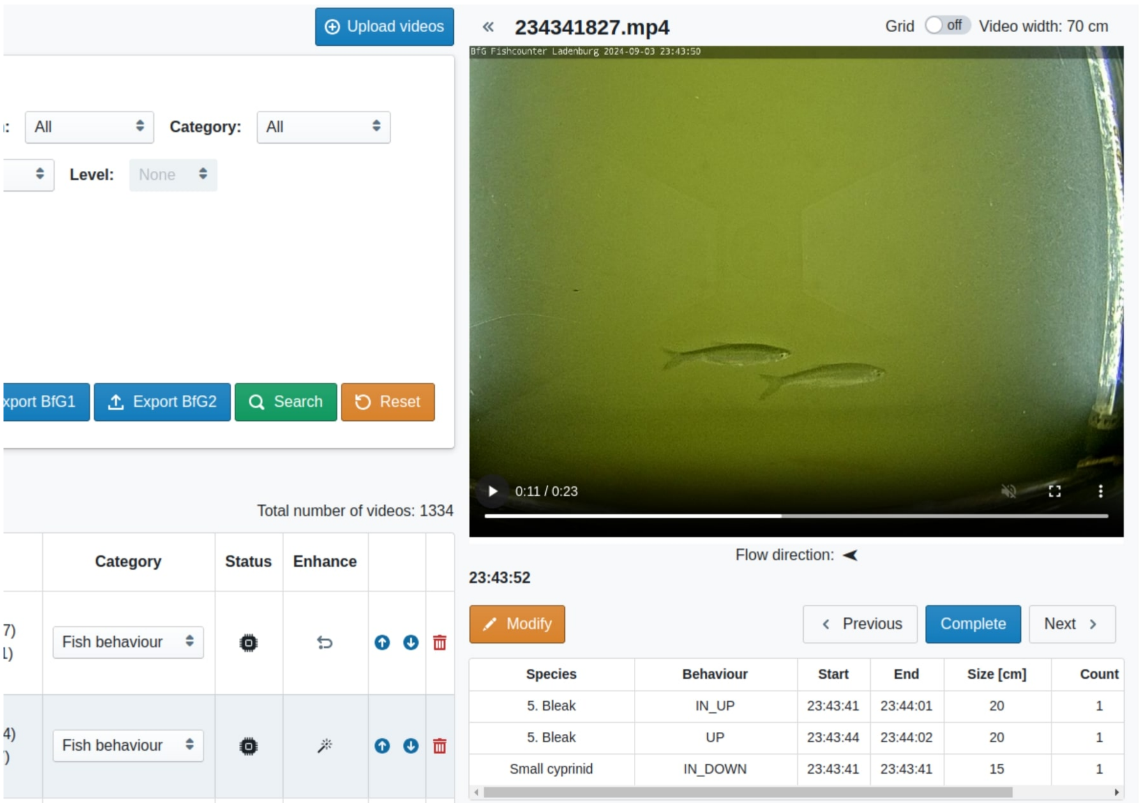
Action: -------- · 0 out of 1334 selected · All: · Total number of videos: 1334

	Video	Date	Location	Environmental conditions	Category	Status	Enhance	
	235155574.mp4	2024-09-01	River: Neckar Site: Ladenburg 2 Counter: counter Camera: Single	Turbidity: Low (0.97) Low Light: None (1)	Fish behaviour			

Figure 1. Main application view with all the primary functionality.

In the application, it is also possible to open an expandable view by selecting the name of the video of interest, as shown in Figure 2. This view contains a video player and details on fish species, behavior, occurrence times, size, count, and whether the fish appear to have visible external injuries. The user can verify or modify this information and delete individual data rows.

Before this thesis work, the application could only run the environmental conditions and the FNF (Fish/No-Fish) modules. The application saved the results of the environmental conditions module in the DB (Database) and displayed them in the main application view. The FNF module could detect fish but did not save any results. It stopped processing as the first fish was detected, offering only "fish" and "no fish" classifications. Fish counting and identifying the species, size, and behavior had to be performed manually.



23:43:52

Species	Behaviour	Start	End	Size [cm]	Count
5. Bleak	IN_UP	23:43:41	23:44:01	20	1
5. Bleak	UP	23:43:44	23:44:02	20	1
Small cyprinid	IN_DOWN	23:43:41	23:43:41	15	1

Figure 2. Expandable video view with fish counts.

3.3 Functional Requirements

Functional requirements specify the desired behavior and functions of the system [14]. During the requirement analysis, the author, in collaboration with BfG and TalTech's Center for Environmental Intelligence and Sensing, defined the following functional requirements:

- The application must allow the running of each processing module independently and in conjunction with other modules.
- The application must allow for batch processing of the videos, which will not be interrupted by a processing error in individual videos.
- The FNF module must not stop when the first fish is detected.
- The FNF module must classify videos as "fish," "no fish," and "fish swarm" videos.
- The FNF module must be able to obtain fish bounding-box images and tracking data.
- The results of the FNF module must be saved in the DB for later use in other modules.
- The behavior module must be able to classify videos as "fish behavior".
- The application must save the behavior evaluation results to the DB.

- The application must save the size estimation results to the DB.
- The application must save the species prediction results to the DB.
- Users should be able to map user-defined fish species labels to the labels used by the species module.
- The application must allow the users to override the results of the modules, preserving both user-defined data and module results in the DB.
- The application must determine whether the modules or a user saved the fish data and display the correct one; user data always has priority over the modules' data.

3.4 Non-Functional Requirements

Non-functional requirements define how the system should work to achieve the functional requirements, considering performance, security, reliability, and scalability. Below are the defined non-functional requirements with a focus on computational efficiency.

- The application must use the most recent applicable Python and library versions to ensure high reliability and optimal execution time.
- The application must utilize the GPU (Graphics Processing Unit) to perform the prediction steps on the systems where the GPU is available.
- If the user starts the processing with multiple modules, video frame extraction must be performed only once at the beginning of the processing workflow; the frames must be kept in memory to be reused in later processing stages.
- If the user starts the processing with multiple modules, the bounding-box images must be prepared and saved to storage only once during the processing workflow; the image paths must then be kept in memory to be reused in later processing stages.
- If the user starts the processing with multiple modules, the FNF module results must be saved in the DB and also kept in memory, and reused in later processing stages.
- If the user starts the processing with multiple modules, the size module results must be kept in memory and reused in the species module.
- The application must use threading or multiprocessing in frame extraction and bounding-box image preparation steps.
- The processing time of a single three-minute video must be thirty or fewer seconds.

4 Development Work

This chapter describes the development work performed as part of this bachelor's thesis. The main contributions of this thesis to the development work are described in the following sections.

4.1 Development Preparation

During the development preparation phase, the author performed the work required to further develop the Smart Fish Counter Application. The preparations included installing the application on the author's laptop and obtaining and examining a special preannotation script along with the ML models to be integrated into the application.

Although the Smart Fish Counter Application uses Docker to containerize its different parts, it is intended for use in Linux-based operating systems and has been developed to work under the 22 version of Ubuntu. To be able to start the development work, the author set up the dual boot with the Windows 11 operating system already installed on his laptop and Ubuntu 22.04.5. After the successful dual boot setup, the author installed the application on his laptop, performed its initial configuration, and examined its functionality and the code base to familiarize himself with its structure, components, and technology.

The TalTech's Center for Environmental Intelligence and Sensing also provided the author with the pre-trained ML models that must be integrated into the application and used in the behavior, size, and species modules, along with the preannotation script previously used to run all the models in sequence to analyze the fish videos locally. The script is a complex pipeline with various processing steps consisting of almost 4,000 lines of code. The author thoroughly analyzed the script and its different parts and planned its refactoring and adaptation to the Smart Fish Counter Application, considering the application's structure and required modifications to run the processing modules as independently as possible.

4.2 DB Updates

After the preparation work was completed, the DB structure was updated to accommodate the new modules. As Appendix 2 shows, several columns were added to the videos_fish table, and a videos_fishlabelmapping table was created to satisfy the requirement that users should be able to map user-defined fish species labels to those used by the species module.

The author also identified the need to create a new Django migration to alter the schema of the videos_fish table, specifying the default values of the columns. Appendix 3 shows the migration for the videos_fish table column defaults.

4.3 UI Updates

Before the work on this bachelor's thesis, a drop-down menu was used to initiate processing. As shown in Figure 3, it contained options to delete and download videos and to start environmental conditions and FNF modules independently and together.

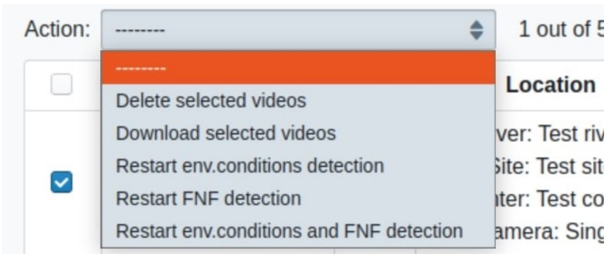
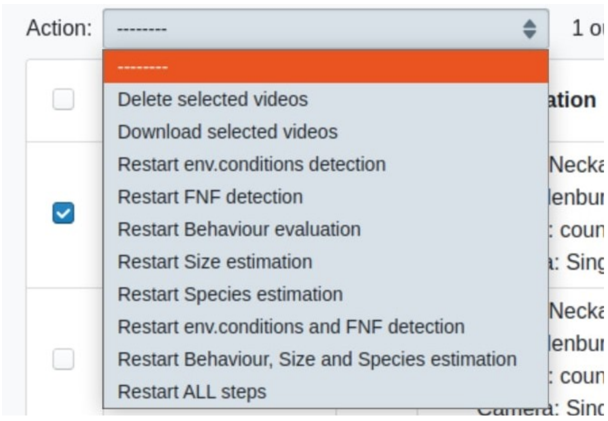


Figure 3. Initial action drop-down menu.

With the integration of the new modules, the menu was updated, as shown in Figure 4.


Action:

Delete selected videos
Download selected videos
Restart env.conditions detection
Restart FNF detection
Restart Behaviour evaluation
Restart Size estimation
Restart Species estimation
Restart env.conditions and FNF detection
Restart Behaviour, Size and Species estimation
Restart ALL steps

Figure 4. Updated action drop-down menu.

Five new options were added to launch the behavior, size, and species modules: three options to launch them independently, an option to launch all three new modules, and an option to launch all five modules available in the application. This update covers the functional requirement of allowing each processing module to run independently or in conjunction with other modules. The author also updated the logic behind the menu to start selected actions in the back-end.

4.4 FNF Module Updates

Several updates were performed in the FNF module before integrating the three new modules. First, the detection model was updated from YOLOv5 (You Only Look Once v5) 6.1 to the 7th version to make the FNF module consistent with the preannotation script. Afterward, the module was adjusted according to the following functional requirements:

- The FNF module must not stop when the first fish is detected.
- The FNF module must classify videos as "fish," "no fish," and "fish swarm" videos.
- The FNF module must be able to obtain fish bounding-box images and tracking data.
- The results of the FNF module must be saved in the DB for later use in other modules.

To start with, the author changed the detection logic to satisfy the requirement that the module is not to be stopped as soon as the first fish is detected. The module could now go through all the video frames and run the detection model for each frame. This update was a substantial change because the obtained YOLO detections are needed in other modules.

To meet the requirement of classifying the videos as "fish," "no fish," and "fish swarm" videos, the author integrated two helper ML models from the preannotation script. Both models represent neural networks that draw conclusions based on the input that is fed to them. The logic for preparing the models' input from the detection data was taken from the preannotation script, refactored, and adapted for the application. The first integrated model determines if the video contains fish and is used in addition to the existing classification, minimizing the number of videos falsely classified as "fish" videos. The second model determines whether the video contains a fish swarm or individual fish and is used to classify the videos as "fish swarm" videos if detected.

To satisfy the third requirement of obtaining fish bounding-box images and tracking data in the FNF module, the author updated and adapted the logic of the preannotation script for the bounding-box image extraction and tracking data retrieval into the application. The implemented logic allowed the extraction of bounding-box images using the video frames and previously retrieved detections, saving them into internal storage, holding the image paths in memory, and deleting the images after the processing was finished.

Finally, the author performed analysis and development work to meet the requirement of saving the FNF module results to the DB for later use in other modules. Several options were considered, including storing each individual detection as a DB row and finding a way to store consolidated data, and it was decided to use Pandas DataFrames as the module's output, allowing effective manipulation and storage of large amounts of data [15]. The output comprises YOLO model detections merged with the tracking data and saved in internal storage, whose reference is held in the DB. If processing was started with multiple modules, this output is also stored in memory and reused in later processing stages.

4.5 Behavior Module

During the integration of the behavior module, the author reviewed and updated the complex logic used to determine the behavior of the fish in the preannotation script, which consisted of over 600 lines of code. As a result of the code refactoring and optimization, the code base was reduced by half, with a final module's length of 291 lines of code. The module uses the previously obtained tracking data, which is passed directly to it if the processing was started with multiple modules or retrieved from the DB.

To meet the requirement of classifying the videos as "fish behavior", a new logic was added to the module that checks its output. Additionally, new functionality was implemented to satisfy the requirement of saving behavior evaluation results to the DB.

4.6 Fish Size Module

Several steps were taken to integrate the fish size module into the application. Firstly, since the module required the paths to the previously obtained fish bounding-box images, the author added the logic of forwarding those paths to the module along with the tracking data passed to the module in the same way as in the behavior module. Secondly, the author adapted the logic of preparing the size estimation dataset and obtaining the predictions from the preannotation script. The size is estimated by utilizing three helper ML models:

- A model for retrieving embeddings from the fish bounding-box images.
- A model for estimating the fish size based on the previously retrieved embeddings.
- A model for correcting the initial size estimation to improve its accuracy.

To meet the requirement of saving the size estimation results in the DB, they are combined with fish tracking data, and the obtained result is saved in the existing fish row in the DB.

4.7 Fish Species Module

The last module integrated into the application was the fish species module, for which the author also used the logic of the preannotation script. The module required the same input as the fish size module; therefore, the fish bounding-box images and tracking data were also transmitted to this module.

The module also required the fish size data to correct the initial species estimations based on the maximum and minimum species sizes from the biological studies, and for that, saving the size data was adjusted to hold it in memory and reuse it in the species module when the processing is started for both the size and species modules. Additionally, to satisfy the requirement of saving the species prediction results to the DB, they are combined with fish tracking data, and the obtained result is saved in the existing fish row in the DB.

4.8 Additional Updates

After all the modules were successfully integrated, a few additional updates were performed to meet the following three functional requirements:

- The application must allow for batch processing of the videos, which will not be interrupted by a processing error in individual videos.

- The application must allow the users to override the results of the modules, preserving both user-defined data and module results in the DB.
- The application must determine whether the modules or a user saved the fish data and display the correct one; user data always has priority over the modules' data.

To meet the first requirement, the application's batch-processing functionality was updated, introducing the ability to process videos uninterruptedly and skip those with errors.

Afterwards, updates were performed to meet the requirement of allowing users to override the modules' results while preserving both user-defined data and module results in the DB. A helper method was created in the back-end triggered the first time the user updated the fish data and used to sync AI data with the fish data fields not updated by the user.

Finally, the author updated the back-end's fish data serializer to forward the correct data to the front-end, meeting the last requirement of displaying the correct fish data depending on whether the user-defined data are available. For this update, the last_modified_by column of the videos_fish table was used, defined when the user saves fish data.

4.9 Testing and Validation

During the development work, the author continuously tested and validated the integrated functionality using the local installation of the Smart Fish Counter Application and its installation on the staging environment. The staging environment is hosted in TalTech HPC and provides the resources needed to test the application under real-life conditions.

For testing purposes, the author was given 30 videos of different lengths with single fish, fish swarms, and not containing any fish. At the beginning of the development work, the author processed those videos with the preannotation script and saved the processing results for later comparison with the results obtained from the application.

In addition to the author's testing, the supervisor and BfG representatives also conducted testing rounds and shared their feedback on the integrated functionality. This feedback cycle allowed for the timely identification of issues and bugs, which the author later fixed.

5 Optimization Work

This chapter describes the second main objective of this bachelor's thesis: optimizing the application's ML algorithms to improve computational efficiency and reduce processing times. Information on the optimization goal, specific optimization methods used by the author, and optimization results are presented in the following sections.

5.1 Optimization Goal

The main optimization goal of this thesis work was defined as a non-functional requirement:

- The processing time of a single three-minute video must be thirty or fewer seconds.

Meeting this goal is crucial for the effective operation of the Smart Fish Counter Application as the thirty-second objective represents a processing speed roughly three times faster than that of a human rater, based on the statistics gathered by TalTech from eight human raters of more than 400,000 videos. This objective would, therefore, allow for a single instance of the application to replace three human raters, justifying any additional costs required for the initial setup and maintenance of the system by end-users.

5.2 Optimization Preparation

During the optimization preparation phase, the author prepared a balanced sample of fish videos to test different optimizations and added functionality to measure the execution time of each module and the individual processing steps with millisecond precision.

To prepare a balanced sample of fish videos for optimization testing, the author used the 400,000 human-annotated fish videos provided by TalTech's Center for Environmental Intelligence and Sensing, recorded at three different BfG sites in Germany in different rivers. The video durations differed, but approximately one-quarter of the videos were three minutes long. The author examined more than 5,000 human-annotated fish videos and carefully selected three-minute ones to represent the worst-case scenario for human raters, which is to watch the longest-duration videos.

The selected sample included 50 videos: 25 with a single fish and 25 with fish swarms. All selected videos have a frame rate of 5 FPS (Frames Per Second); 16 videos have a resolution of 1280 x 960 pixels, and the remaining 34 have a resolution of 800 x 600 pixels. The difference in resolutions is due to the different cameras used at the BfG sites.

After preparing the balanced sample, the author implemented the logic for measuring the execution time of each module and processed the selected videos on the staging server hosted in TalTech HPC to determine their initial average processing times, which were later used as a baseline for further optimizations. Figure 5 shows the initial average processing times, with the following numbers for different modules:

- Environmental conditions module: 53.24 seconds.
- FNF module: 150.96 seconds.
- Behavior module: 0.08 seconds.
- Size module: 3.79 seconds.
- Species module: 8.20 seconds.
- Total: 216.28 seconds.

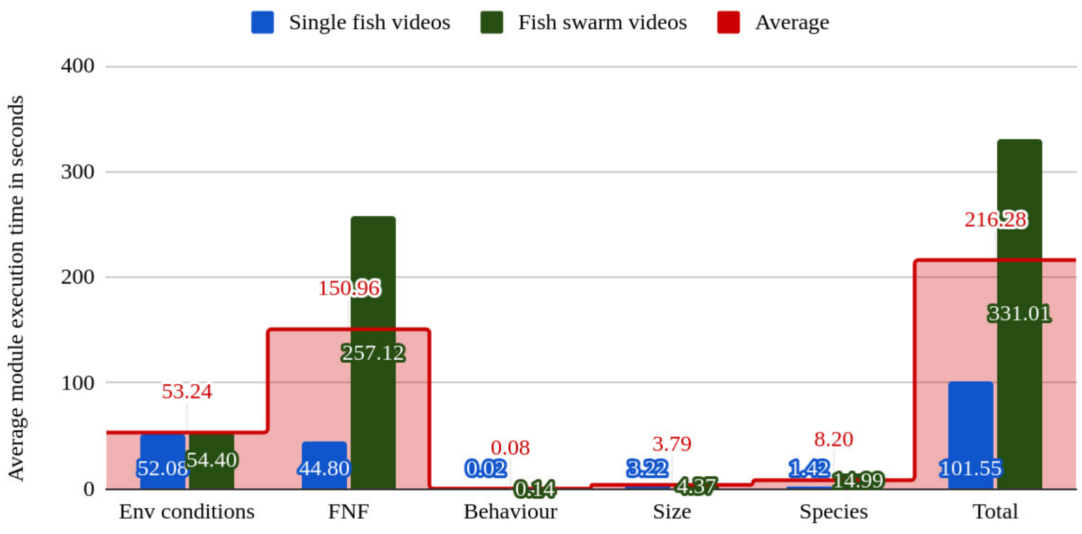


Figure 5. Initial average processing times.

5.3 Python, Libraries, and Environmental Conditions Updates

After the preparation work was completed, the author started the work on the first optimization. Possible updates to the Python and library versions were analyzed, as utilizing the most recent versions ensures reliability and performance. As a result of this analysis, the Python version was updated from 3.8 to 3.10 (the latest version compatible with some utilized libraries), and numerous libraries were updated to the latest compatible versions.

In addition, the author also performed updates in the environmental conditions module. Before the work on this bachelor's thesis, the environmental conditions module used the Keras library, which is used in Python to work with neural networks, to load and use the models for determining environmental conditions levels. To bring all prediction logic to a common form, TalTech's Center for Environmental Intelligence and Sensing retrained environmental condition models using the PyTorch library, which is also used in the FNF, size, and species modules, and adapted those new models in the preannotation script. The author also performed a similar adaptation to load and use the new environmental condition models with PyTorch in the application for consistency and performance reasons.

As a result of this optimization, the total average processing time was reduced by 31.88% compared to the initial total average. Figure 6 shows the first optimization results.

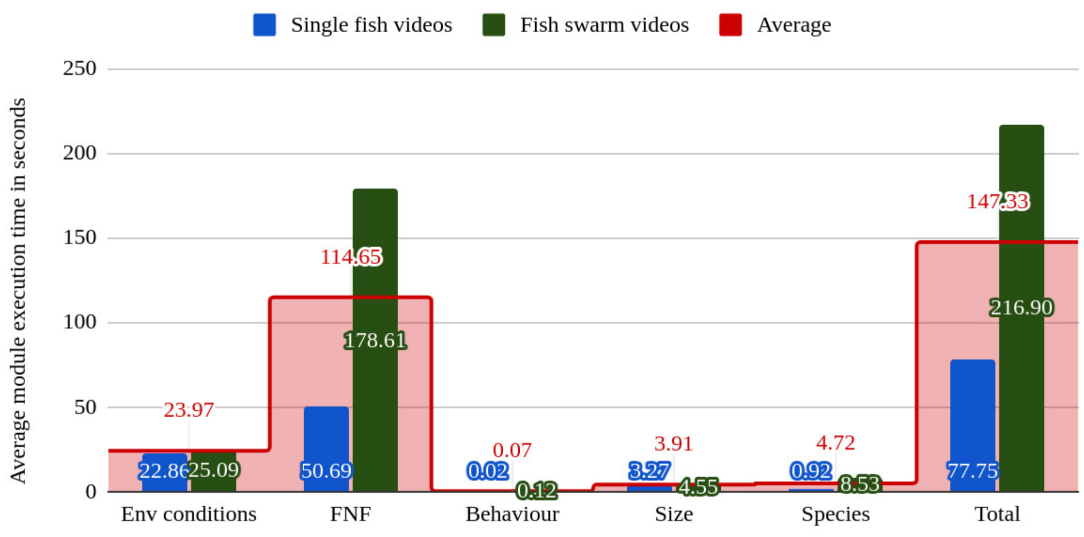


Figure 6. Average processing times after the first optimization.

As can be seen in the figure, the average processing times after the first optimization were:

- Environmental conditions module: 23.97 seconds.
- FNF module: 114.65 seconds.
- Behavior module: 0.07 seconds.
- Size module: 3.91 seconds.
- Species module: 4.72 seconds.
- Total: 147.33 seconds.

5.4 Introduction of GPU Usage

During the second optimization, the author focused on introducing GPU usage into the application. The use of the GPU provides faster processing times for ML applications than that of the CPU (Central Processing Unit), because these applications require large amounts of data to be processed in parallel, and GPUs are specifically designed for parallel processing [16].

To use the GPU in the application, the author first updated the configuration of the Django Docker container. Afterward, a new logic was introduced into the application, which allows loading the models to obtain environmental conditions, fish object detections, fish bounding-box image embeddings, and species to the GPU via the PyTorch library.

The introduction of GPU usage resulted in a 71.07% improvement in total average processing time compared to the first optimization's total average and an 80.29% improvement compared to the initial total average. Figure 7 shows the average processing times after the second optimization, with the following numbers for different modules:

- Environmental conditions module: 16.10 seconds.
- FNF module: 22.15 seconds.
- Behavior module: 0.08 seconds.
- Size module: 3.39 seconds.
- Species module: 0.90 seconds.
- Total: 42.63 seconds.

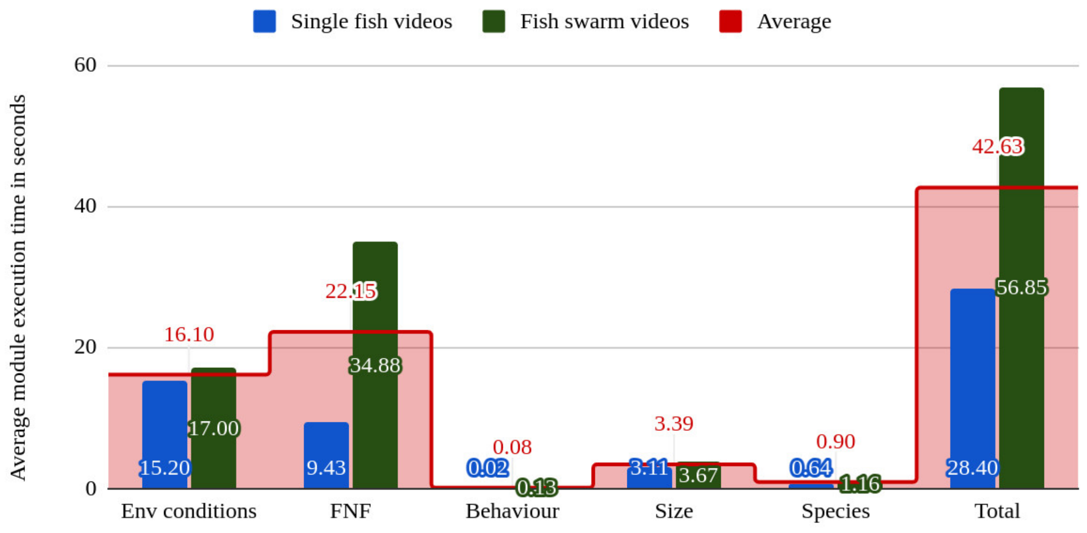


Figure 7. Average processing times after the second optimization.

5.5 Video Frame Extraction Optimization

After the first two optimizations, the author began to optimize frame extraction. Initially, the application extracted the frames eight times, mainly because each module was developed independently: once for each of the five environmental conditions and once in the FNF, size, and species modules. The author adjusted this to retrieve the frames only once at the beginning of the processing workflow, hold them in memory, and reuse them for each of the mentioned processing steps.

These changes improved the total average processing time by 30.46% compared to the previous optimization and by 86.29% compared to the initial total average. Figure 8 shows the average processing times after the third optimization, with the following numbers:

- Environmental conditions module: 3.95 seconds.
- FNF module: 21.23 seconds.
- Behavior module: 0.09 seconds.
- Size module: 3.46 seconds.
- Species module: 0.92 seconds.
- Total: 29.65 seconds.

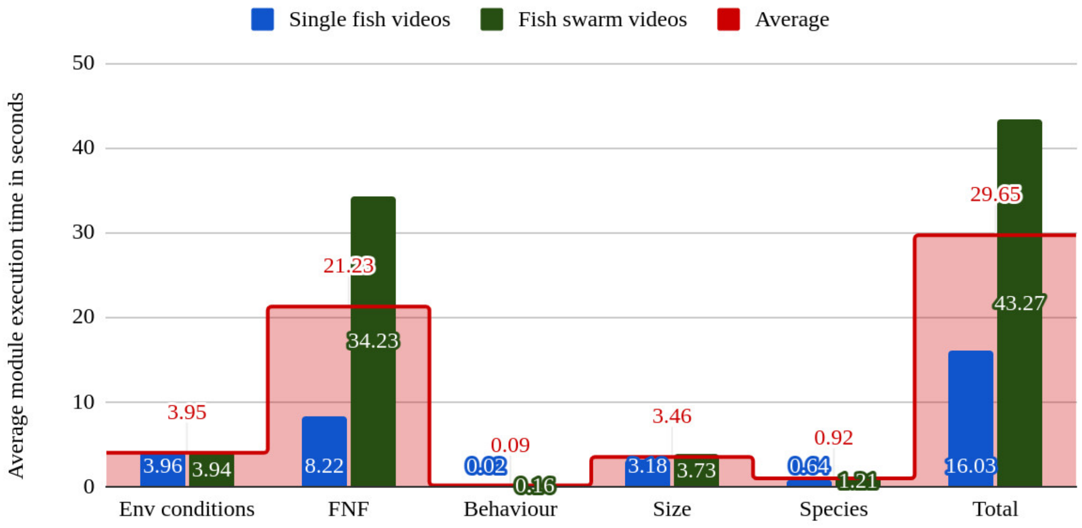


Figure 8. Average processing times after the third optimization.

5.6 Multiprocessing and Threading

The last optimization on which the author worked was the use of multiprocessing and threading. The analysis was performed to determine which parts of the processing workflow could be optimized and whether multiprocessing or threading would best suit their specifics. This analysis showed that the most suitable processing workflow steps for multiprocessing and threading are video frame extraction and fish bounding-box image preparation due to their iterative nature.

The author had to consider several nuances when determining whether to utilize multiprocessing or threading for the selected processing steps. First of all, in Python, there is a lock called GIL (Global Interpreter Lock) which restricts the usage of threading. The GIL allows only one thread to control the interpreter at any given time, and it is needed due to the specifics of Python's memory management [17]. Since the GIL prevents the threads from running simultaneously, they could only be run concurrently. However, several Python libraries and frameworks, such as NumPy and OpenCV, can release the GIL to run their functions. Additionally, the specifics of Python's threading and multiprocessing implementations dictate their application areas: threading is a better option for IO (Input/Output)-bound tasks and multiprocessing is for CPU-bound tasks [18].

As a result of the analysis, the author chose threading for both frame extraction and fish bounding-box image preparation. In the case of frame extraction, threading usage is conditioned by the fact that the frame extraction is performed via the OpenCV library, which can release the GIL. Additionally, there are several issues with OpenCV's multiprocessing, which significantly complicates its implementation. In the case of fish bounding-box image preparation, threading is preferable to multiprocessing because this task involves saving the images to internal storage, which is an IO-bound task.

The use of threading for frame extraction and fish bounding-box image preparation improved the total average processing time by 8.76% compared to the previous optimization and by 87.49% compared to the initial total average. Figure 9 shows the average processing times after the fourth optimization, with the following numbers for different modules:

- Environmental conditions module: 3.31 seconds.
- FNF module: 19.21 seconds.
- Behavior module: 0.08 seconds.
- Size module: 3.48 seconds.
- Species module: 0.96 seconds.
- Total: 27.05 seconds.

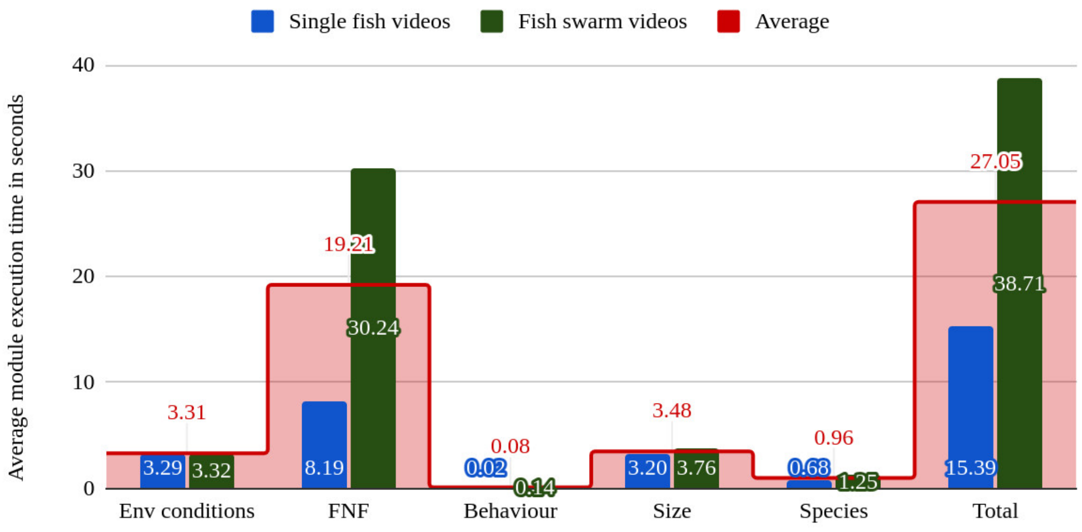


Figure 9. Average processing times after the fourth optimization.

5.7 Optimization Results Summary

During the optimization phase of this bachelor's thesis, the author performed four different optimizations to achieve the non-functional requirement of processing a single three-minute video in thirty seconds or fewer. Table 2 shows the initial average processing times and the optimizations' results, including the average processing times for each module and the total average processing times in seconds with their respective percentage gains from the initial times.

Table 2. Average processing times comparison.

	Env. conditions	**FNF**	**Behavior**	**Size**	**Species**	**Total**
Initial times	53.24	150.96	0.08	3.79	8.20	216.28
First optimization	23.97 *(54.98%)*	114.65 *(24.05%)*	0.07 *(10.38%)*	3.91 *(-3.07%)*	4.72 *(42.41%)*	147.33 *(31.88%)*
Second optimization	16.10 *(69.76%)*	22.15 *(85.32%)*	0.08 *(2.32%)*	3.39 *(10.58%)*	0.90 *(89.02%)*	42.63 *(80.29%)*
Third optimization	3.95 *(92.59%)*	21.23 *(85.94%)*	0.09 *(-13.41%)*	3.46 *(8.94%)*	0.92 *(88.75%)*	29.65 *(86.29%)*
Fourth optimization	3.31 *(93.79%)*	19.21 *(87.27%)*	0.08 *(-0.05%)*	3.48 *(8.28%)*	0.96 *(88.24%)*	27.05 *(87.49%)*

As the table shows, during the course of the four optimizations, the initial total average processing time was reduced from 216.28 to 27.05 seconds, which is an improvement of 87.49% or eight times. The goal of reducing the processing time of a single three-minute video to thirty seconds or fewer was achieved.

6 Summary

The objectives of this bachelor's thesis were:

- Integration of computer vision ML algorithms for the automated detection of fish species, size class, and individual fish behavioral evaluation;
- Testing and validation of the integrated algorithms;
- Optimization of the application's ML algorithms to improve computational efficiency and reduce processing times.

Multiple steps were taken to achieve the first objective, including comparing the proposed solution with the existing ones, providing a technology overview, analyzing the functional requirements, and conducting development work. During the development work, in addition to integrating the new modules, the existing modules and other application parts were also updated to accommodate the integrated functionality.

The second objective was achieved by multiple means, such as testing cycles conducted by the supervisor, having meetings and obtaining feedback from the BfG representatives, and continuous testing and validation of the integrated functionality performed by the author using the application's local installation and its installation on the staging environment.

The third objective was achieved by analyzing the non-functional requirements and using them for the optimization work. The main non-functional requirement of this thesis was to reduce the processing time of a single three-minute video to thirty seconds or fewer. This requirement was successfully met, and the average processing time was reduced from 216.28 seconds to 27.05 seconds, which represents an 87.49% improvement.

The Smart Fish Counter Application was successfully updated per this work's objectives. The next steps in the application development involve further enhancing the ML models that were already pre-trained before this bachelor's thesis and provided to the author and adding an analytics UI to enable users to post-process the results.

References

[1] J. Soom, V. Pattanaik, M. Leier, and J. A. Tuhtan, “Environmentally adaptive fish or no-fish classification for river video fish counters using high-performance desktop and embedded hardware,” *Ecological Informatics*, vol. 72, 2022. DOI: 10.1016/j.ecoinf.2022.101817.

[2] K. Schwaber and J. Sutherland, The 2020 Scrum Guide, Accessed: Nov. 17, 2024. [Online]. Available: https://scrumguides.org/scrum-guide.html.

[3] “Technical, Logistical, and Economic Considerations for the Development and Implementation of a Scottish Salmon Counter Network,” *Scottish Marine and Freshwater Science*, vol. 7, no. 2, 2016.

[4] Riverwatcher | Fish Monitoring for Rivers, Accessed: Nov. 13, 2024. [Online]. Available: https://www.riverwatcher.is.

[5] VAKI Riverwatcher - FISHBIO | Fisheries Consultants, Accessed: Nov. 13, 2024. [Online]. Available: https://fishbio.com/technology/riverwatcher.

[6] Optical Simsonar FishCounters for monitoring the migration of fish, Accessed: Nov. 27, 2024. [Online]. Available: https://www.simsonar.com.

[7] TiVa FC Product Sheet, Accessed: Nov. 27, 2024. [Online]. Available: https://tiva.se/wp-content/uploads/2024/03/2024-product-sheet-fc.pdf.

[8] Fish algorithms and AI | TiVa, Accessed: Nov. 27, 2024. [Online]. Available: https://tiva.se/en/fish-ai.

[9] Autonomous Fish Monitoring Systems and Fish Counting Equipment - Biotactic Inc. Accessed: Nov. 17, 2024. [Online]. Available: https://www.biotactic.com/fish-monitoring-fish-counting-services.

[10] I AM HydroCam EN flyer, Accessed: Nov. 27, 2024. [Online]. Available: https://iamhydro.com/wp-content/uploads/2024/04/i-am-hydro_flyer_hydrocam_2021_en.pdf.

[11] S16 DualFlex | MOBOTIX AG, Accessed: Nov. 27, 2024. [Online]. Available: https://www.mobotix.com/en/products/outdoor-cameras/s16-dualflex.

[12] S. Susnjara and I. Smalley, What Is Docker? | IBM, Accessed: Dec. 01, 2024. [Online]. Available: https://www.ibm.com/topics/docker.

[13] How Ansible works | Ansible Collaborative, Accessed: Dec. 01, 2024. [Online]. Available: https://www.ansible.com/how-ansible-works.

[14] Functional vs. Non Functional Requirements - GeeksforGeeks, Accessed: Dec. 03, 2024. [Online]. Available: https://www.geeksforgeeks.org/functional-vs-non-functional-requirements.

[15] Pandas - Python Data Analysis Library, Accessed: Dec. 06, 2024. [Online]. Available: https://pandas.pydata.org.

[16] FPGA vs. GPU vs. CPU – hardware options for AI applications | Avnet Silica, Accessed: Dec. 11, 2024. [Online]. Available: https://my.avnet.com/silica/resources/article/fpga-vs-gpu-vs-cpu-hardware-options-for-ai-applications.

[17] A. Ajitsaria, What Is the Python Global Interpreter Lock (GIL)? – Real Python, Accessed: Dec. 13, 2024. [Online]. Available: https://realpython.com/python-gil.

[18] Difference Between Multithreading vs Multiprocessing in Python - GeeksforGeeks, Accessed: Dec. 13, 2024. [Online]. Available: https://www.geeksforgeeks.org/difference-between-multithreading-vs-multiprocessing-in-python.

Appendix 1 – Non-Exclusive License for Reproduction and Publication of a Graduation Thesis[1]

[1]The non-exclusive license is not valid during the validity of access restriction indicated in the student’s application for restriction on access to the graduation thesis that has been signed by the school’s dean, except in case of the university’s right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive license, the non-exclusive license shall not be valid for the period.

Appendix 2 – New Columns of the videos_fish Table and the Schema of the videos_fishlabelmapping Table

New columns of videos_fish table.

Column	Type	Nullable	Default
behaviour_probability	integer	not null	100
label_probability	integer	not null	100
size_uncertainty	double precision	not null	0
ai_exact_size	double precision	not null	5
track_id	double precision	not null	-1

videos_fishlabelmapping table schema

Column	Type	Nullable	Default
id	integer	not null	nextval('videos_fishlabelmapping_id_seq' ::regclass)
name	character varying(255)	not null	""
created_at	timestamp with time zone	not null	timezone('utc', now())
updated_at	timestamp with time zone	not null	timezone('utc', now())
label_id	integer		null

Appendix 3 – Django Migration for the videos_fish Table Column Defaults

```
from django.db import migrations


class Migration(migrations.Migration):

    dependencies = [
        ('videos', '0014_fish_tracking_id'),
    ]

    operations = [
        migrations.RunSQL(
            """
            ALTER TABLE videos_fish
            ALTER created_at SET DEFAULT now(),
            ALTER updated_at SET DEFAULT now(),
            ALTER user_behaviour SET DEFAULT 'UNDEFINED',
            ALTER user_end SET DEFAULT '00:00:00',
            ALTER injured SET DEFAULT false,
            ALTER user_size SET DEFAULT 5,
            ALTER user_start SET DEFAULT '00:00:00',
            ALTER user_count SET DEFAULT 1,
            ALTER ai_behaviour SET DEFAULT 'UNDEFINED',
            ALTER ai_count SET DEFAULT 1,
            ALTER ai_end SET DEFAULT '00:00:00',
            ALTER ai_size SET DEFAULT 5,
            ALTER ai_start SET DEFAULT '00:00:00',
            ALTER behaviour_probability SET DEFAULT 100,
            ALTER label_probability SET DEFAULT 100,
            ALTER size_uncertainty SET DEFAULT 0,
            ALTER ai_exact_size SET DEFAULT 5,
            ALTER tracking_id SET DEFAULT -1;
            """
        ),
    ]
```

Appendix 4 – Acknowledgments

I, Aleksei Lutkov, the author of this bachelor's thesis, would like to thank my supervisor, Jeffrey A. Tuhtan, for his assistance and support during the writing of this thesis.

I would also like to thank Aleksandr Ivanov and Alexandra Kolosova from TalTech's Center for Environmental Intelligence and Sensing for their help and guidance throughout this thesis work.

It has been a great pleasure working with you on the Smart Fish Counter Application and contributing to the development of such an important tool that can help preserve freshwater vertebrate populations and impact the future of sustainable fisheries.