Jan 2, 2024

Using computer vision and AI techniques to improve crowd safety management

Final report, BEng Software Technology, 2020-2024

Anton Irvold
University of Southern Denmark, Technical Faculty
anirv20@student.sdu.dk
Author

Christoffer Krath
University of Southern Denmark, Technical Faculty
chkra19@student.sdu.dk
Author

Sune Lundø Sørensen
University of Southern Denmark, MMMI
slso@mmmi.sdu.dk
Supervisor

Mikkel Baun Kjærgaard
University of Southern Denmark, MMMI
mbkj@mmmi.sdu.dk
Supervisor

Christian Sejlund
Event Safety (Smukfest & Muskelsvindfonden)
christian@eventsafety.dk
Company Collaborator

Sofie Dahl
Event Safety (Smukfest & Muskelsvindfonden)
sofie@eventsafety.dk
Company Collaborator

ABSTRACT

In spring of 2023 we reached out to Event Safety A/S regarding the possibility of a collaboration with them regarding our final project of our education: B.E in software technology.
The idea was a software application implementing artificial intelligence and computer vision to help improve crowd safety management. Following an agreement to collaborate and various festival invitations by Event Safety to collect data, a problem statement was produced: *Can computer vision software and AI techniques be leveraged to improve crowd overview for security guards, by receiving video feed from large crowds, and ultimately improve crowd safety?*

The projects structure and planning followed a waterfall model for the purposes of development. Along with this, we had predefined meetings arranged with our collaborators from Event Safety to showcase progress and gather feedback for further development. When supervision was required, we turned to our supervisor and scheduled a meeting, usually within a couple of days to quickly move on.

Research was conducted into the academic litterature on computer vision and existing crowd counting methods, leading to the usage of the SASNet crowd counting deep learning model. After implementing the model successfully, the focus of the project shifted to image manipulation for better results. This includes, but is not limited to, perspective correction, image upsampling and image downsampling. Backend implementation of the model was run on UCloud - a cloud computing service hosted by SDU - and a frontend developed in Python for application of heatmaps and the ability to segment the crowd based on user drawn polygons. When development of the project had concluded, validation and verification commensed. Manual count comparisons, and thourough user test through a focus group was held for employees at Event Safety to gather final feedback and perspectives on the project and its viability.

Based on the presentation and feedback, we are able to conclude that we succeeded in implementing computer vision and AI techniques in a crowd management system. Comparing the feedback from our collaborators to our initial requirements, the system has adequate precision for situations that makes it useable to the inteded user while allowing for development of more functionality. The system can successfully be leveraged by safety experts to improve crowd safety and crowd safety management

Table of contents

Reading guide

This report is structured chronologically and is therefore meant to be read that way. This being said, it is possible to read the abstract and conclusion and still understand most of the project. It is expected that the reader possesses basic programming and software technology knowledge, specifically Python, to gain deeper insight into the Design and Implementation chapters.

Chapter 1 - Acronyms and definitions, a basic overview of terms that will be used throughout this report.

Chapter 2 - Introduction, will provide background and context for the project.

Chapter 3 - State-of-the-art, will provide insight into what technologies currently exist in the field of computer vision, crowd management and crowd counting.

Chapter 4 - Analysis, highlights the scope, system specification, requirements and risks of this system.

Chapter 5 - Design, goes in-depth with the major design choices made for this system.

Chapter 6 - Implementation, describes and explains the implementation and development of the system itself.

Chapter 7 - Validation and Verification, validates and verifies that the requirements previously set for this system are fulfilled.

Chapter 8 - Operation, describes the thoughts on the operation and maintenance aspects of the project.

Chapter 8 - Discussion, discusses and evaluates our results in relation to the problem statement.

Chapter 9 - Conclusion, concludes the project and our results in relation to our problem statement.

Chapter 10 - Future perspectives, goes through which system features could be developed in the future, and how the project could be expanded and improved.

1 Acronyms and definitions

This section describes the acronyms and definitions used in the report.

Acronyms	Definition
CSMS	Crowd Safety Management System
Crowd Counting	"[…] a privacy-preserving system for estimating the size of inhomogeneous crowds, […] without using explicit object segmentation or tracking" (Chan, Liang, and Vasconcelos 2008) Different from "crowd counting by detection"
ANN	Artificial Neural Network.
CNN	Convolutional Neural Network.
Crowd Surges	A dangerous crowd situation where your movement is controlled by the crowd and not your actions.
Mosh pit	A crowd situation where people vacate an area in the crowd followed by violent or aggressive dancing. The sizes of Mosh pits vary greatly.
Technical user	A developer or technician who can implement the CSMS in the technical setup of a concert or venue.
Non-technical user	A security guard or someone viewing crowd footage through the CSMS without any technical or software-related background.
Choke point	Point with high crowd density and crowd influx.
SASNet	Scale-adaptive selection network Song et al. (2021).
CUDA	Compute Unified Device Architecture (NVIDIA GPU API).
GUI	Graphical User Interface.
LOS	Fruin's Level of Service Fruin (1971).

Table 1: Table of acronyms and definitions used throughout the report.

2 Introduction

This section is going to give an introduction to the domain, reasoning and context of the problem and an introduction to the project collaborators.

2.1 Preface

The 30 ECTS project has been written as the final project for the Bachelor of Engineering in Software Technology at the University of Southern Denmark, Technical Faculty.

A special thanks to our collaborators on this project for making this project possible.

Sune Lundø and Mikkel Kjærgaard for excellent guidance on the project. The expertise has been well-informed and useful throughout the whole semester and helped us navigate the final project.

Event Safety, Sofie Dahl and Christian Sejlund for their time, expertise and knowledge during all phases of the project as well as taking time to teach us about the theory and practice of crowd safety. We would also like to give a big thanks to the rest of Event Safety, Smukfest and Muskelsvindfonden employees for all the feedback we have received.

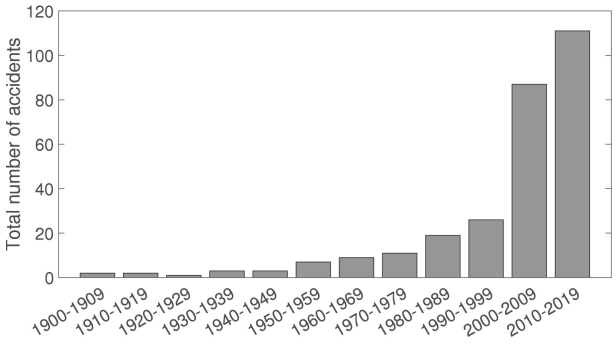
Smukfest and Muskelsvindfonden for inviting us and allowing us to collect data necessary for our project during their festivals and concerts.

Certified drone pilot Mathias Engmark for helping us collect excellent aerial videos during Grøn Koncert.

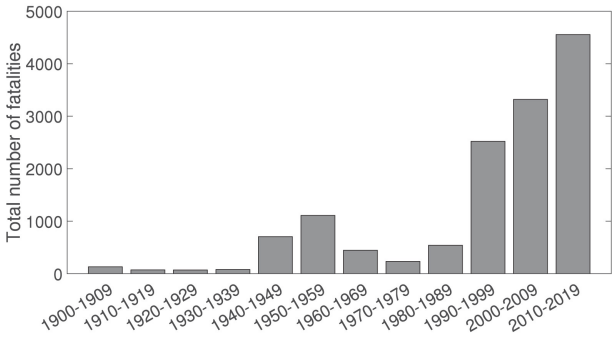
PhaseOne for giving expert feedback on creating useful products that use drone footage.

SDU and TEK for access to camera resources and cloud computing resources.

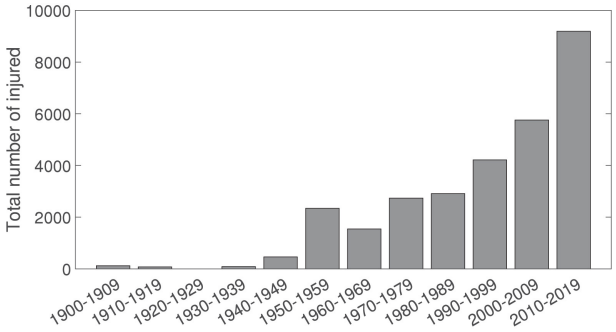
2.2 Context



(a) Accidents by decade



(b) Fatalities by decade



(c) People injured by decade

As seen from **?@fig-fatalities** accidents, fatalities and injuries at festivals and concerts worldwide are on the rise, with just below 5.000 worldwide deaths in the decade of 2010-2019. The increasing trend can be attributed to several causes, according to Raineri (2004): First of all, an increase in the popularity of outdoor music festivals resulting in more and larger festivals with large crowds. Also, high-risk behavior among crowd attendants and the performing artist's music, behavior, and stage show affects the crowd's safety. Cultural influence has also played a large part in the safety of outdoor musical events. This can be behavior such as crowd surfing, moshing, stage diving and the like.

It is worth mentioning that these figures and descriptions of situations are often from media outlets. As such, it is not the full story. While these statistics might make it seem

that general safety and incidents at festivals and the like have gotten worse, it is the opposite. It is generally incredibly safe to attend these events, but using new technologies we will attempt to enhance safety and comfort further.

At the same time, festivals in Denmark and the rest of the world have had a focus on mitigating the safety risk in large crowds. This can be seen by the establishment of the Event Safety Foundation in 2015. This is a joint venture between the Skanderborg Festival Group (Smukfest) and Muskelsvindfonden (Grøn Koncert). This foundation has the purpose of securing the safety of the crowds at these two large Danish festivals, as well as many other events in Denmark. They do this through knowledge sharing, professionals and trained volunteers, courses and counseling. This is the company that this project will be done in collaboration with.

To have the best possible outcome for this project, Event Safety invited us to Grøn Koncert and Smukfest in July and August to see how they work and gather security video footage of real crowds at festivals to use as training data. We will also use Event Safety for their practical and theoretical knowledge of crowd safety to use in the system, user feedback and user testing.

2.3 Problem Statement

Can computer vision software and AI techniques be leveraged to improve crowd overview for security guards, by receiving video feed from large crowds, and ultimately improve crowd safety?

2.4 Planning and method

In the context of developing a software system for our final project, the Waterfall method is adopted as the project management approach. This decision is informed by the specific requirements of the project, including the size of the development group and the constrained timeframe. The Waterfall model, characterized by its linear, sequential and plan-driven approach, offers a structured framework that aligns well with the project's objectives and constraints.

The Waterfall model follows the following approach (Sommerville 2011):

1. Requirement analysis and definition.
2. System and software design.
3. Implementation and unit testing.
4. Integration and system testing.
5. Operation and maintenance: Not part of the scope. The project does, however, deliver documentation and user guides for future use.

This project will largely follow this approach. There are however some exceptions, including the types of tests performed in the implementation and integration phases and the absence of the operation and maintenance phase. These phases would be necessary for the use of our product after the project but are not part of our project scope.

The Waterfall method was chosen for its clear, manageable workflow suitable for a small two-member team compared to an agile Scrum approach, its alignment with the project's timeline for delivering a complex computer vision system, and its structured phases that ensure thorough analysis, design, implementation, and testing.

The project timeline is structured into phases, each aligning with the traditional stages of the Waterfall model as described above by Sommerville (2011). A description of the planning of the project timeline can be seen here as well as in Section 12. The following also includes a description of the deliverables that are part of Sommerville's theory behind the Waterfall method. These will be toned down to reduce the documentation load and make up for the small group size.

1. Preparation Phase (Pre-September): In this initial phase, the groundwork for the project is laid. This includes finalizing the project description, establishing company collaborations, and collecting the necessary data. This phase is critical for ensuring that the project starts with a clear direction and the resources required for success.

2. Analysis and Design (September): The focus in September will be on analyzing the collected data and designing the computer vision system. This stage involves understanding the requirements for crowd safety management and translating these into a viable software design. This phase delivers a requirement specification for the analysis and a software architecture design diagram for the design phase.

3. Implementation (October and November): During these months, the actual development of the software takes place. This phase involves coding the designed system, and ensuring that the software components are implemented correctly to achieve the desired functionality. This phase delivers an implemented system.

4. Testing (Early December): At the beginning of December, the system will undergo testing. This phase is crucial for identifying and rectifying any flaws or inefficiencies in the software. The testing phase ensures that the final product is reliable and meets the project's standards. This phase delivers a documented user test review.

5. Documentation and Reporting: Parallel to the other phases, the project report will be written continuously. This documentation will capture the development process, challenges, solutions, and overall learning from the project. The continuous nature of this task ensures that all aspects of the project are accurately and comprehensively recorded.

Adopting the Waterfall method in this project provides a systematic and predictable framework, crucial for managing the task of integrating computer vision into crowd safety management and being able to deliver a finished proof-of-concept. The structured phases of the Waterfall model align well with the project's timelines and resource allocation, ensuring a cohesive and efficient approach to achieving the project's objectives.

2.5 Stakeholders

"Event Safety A/S" is a key stakeholder in the project. The company, its employees and its volunteers specialize in crowd safety management at large-scale events like festivals, concerts, sporting events and more. Their collaboration includes sharing expertise in crowd safety, assisting in data collection through festival access, and offering feedback for user testing. Event Safety will also provide theoretical and practical insight into crowd safety and invite us to courses where we can learn. Event Safety has an interest in the project's outcome, as it directly aligns with their objective of improving event safety and using new technology to fulfill this need better. The project team is committed to fulfilling the project to Event Safety's needs by integrating their insights, adhering to the responsible use of the data collected on their festivals and ensuring the software system is properly aligned with their interests, practically applicable and effective in actual event scenarios.

3 State-of-the-art

This chapter will describe the state-of-the-art technologies from academic literature regarding computer vision and crowd counting that will be used in this project.

3.1 Convolutional Neural Network (CNN) for Computer Vision

Convolutional Neural Networks (CNN) are a type of Artificial Neural Network (ANN) that are used to "solve difficult image-driven pattern recognition tasks." (O'Shea and Nash 2015) ANN's use input in the form of a multi-dimensional vector, which will be distributed to several hidden layers. The hidden layers will be weighted by evaluating how a stochastic change within itself affects the final output (backpropagation). Deep learning is an ANN with multiple hidden layers. ANN's can be either supervised: Being trained on a set of labeled training data, or unsupervised: Having no labeled training data, but instead minimizing the result of the cost function. According to O'Shea and Nash (2015) and Lempitsky and Zisserman (2010), supervised learning is usually necessary for image-focused pattern recognition tasks.

CNNs are similar to ANNs in almost every way, with the only difference being that CNNs allow encoding image-specific features such as edges, textures, color patterns and shape features. This reduces the complexity of a traditional ANN trained on image data, where a vector representation grows in size exponentially depending on the image size. Reducing the feature vector also reduces the risk of overfitting the model. This is done through the process of convolution. In the convolution layer, local regions of the image vector are scanned for image-relevant features. Usually, the features are run through a pooling layer, reducing its computational complexity by sampling and down-scaling the features. This feature map is then fed to a traditional ANN, with a given amount of hidden layers.

3.2 Solving Scale-Variation with SASNet

Scale variation is one of the main challenges with crowd counting since the perspective in the image will lead to different sizes of heads. One solution to this is proposed by Song et al. (2021). SASNet relies on the fact that heads in a local path share roughly the same scale. It is currently not a common approach to solve scale variation on a continuous scale. SASNet solves it in 5 discrete steps using a weighted average. This bridges the gap between discrete feature extraction and continuous feature extraction. This architecture can be seen in Figure 1.

As can be seen from the diagram, SASNet uses the first 13 convolutional layers from VGG-16 (Simonyan and Zisserman 2014) in the encoder. SASNet then produces 5 feature levels with 5 levels of downsampling (V_n). It produces 5 levels of predictions (P_n). This is what is referred to as the "U-shaped backbone". These are then fed to a confidence branch and a density branch that produces a confidence map and density map for each of the 5 levels. The complete density map is a product of the sum of the individual density and confidence branches using a softmax function for the confidence.

3.3 Leading practical projects of the state-of-the-art crowd management systems

While there are some academic research projects on crowd counting, the practical implementations of this technology are hard to find. In practice, the leading method of crowd counting is a variation of using turnstiles and expertise in knowing what certain densities of people look like and multiplying it with the area of the space, also known as Jacob's method. (Dahl 2023) (Still 2014) Fruin's Level of Service (LOS) is then applied to these

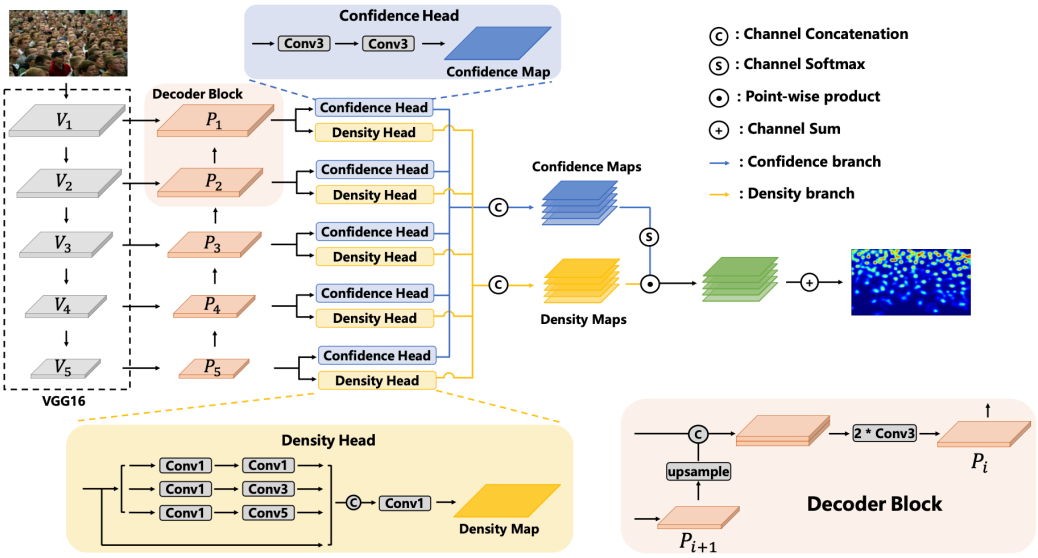


Figure 1: Architecture of the SASNet model (Diagram from Song et al. 2021)

densities. The LOS standard is a measure of how many square meters each person needs to feel comfortable when walking, climbing stairs or standing/queuing. The LOS standard is based on a series of qualitative and quantitative methods (Fruin 1971). Fruin's LOS can be seen in Table 2.

LOS	Density
LOS A	$<$ 0.83 people per sq. meter
LOS B	$<$ 1.07 people per sq. meter
LOS C	$<$ 1.54 people per sq. meter
LOS D	$<$ 3.58 people per sq. meter
LOS E	$<$ 5.38 people per sq. meter
LOS F	$>$ 5.38 people per sq. meter

Table 2: Table of Fruin's Level of Service (LOS). By Fruin (1971). Originally in square feet per person. Converted to densities (people per square meter) with the formula $density = (1/feetperperson) * 10.763915$

Depending on the type of festival Event Safety works with different acceptable LOS. At Grøn which is a more family-friendly festival with people sitting an acceptable service level might be LOS D close to the stage and LOS C farther from the stage. At Smukfest an acceptable LOS might be LOS D or near LOS E in some places. LOS F is usually never acceptable as this can quickly become a safety issue. (Dahl 2023)

According to the best of our collaborators at Event Safety's knowledge, there have not been any real projects that aim to use technology for crowd counting and density estimations at festivals and concerts in Denmark, nor any large-scale projects in Europe. The only example they know of was a project at Roskilde Festival that tried to use cell data to generate a heatmap of crowd densities and crowd size. This project was abandoned because of its inaccuracy.

Commercial applications for object-detection-based cameras are implemented at some places like airports, malls, sidewalks and supermarkets to manage queues, flow and count small numbers of people. This is products like Hikvision, TeraBee, Haltian, FootfallCam. We have not been able to find projects that use cameras to create a detailed heatmap of densities and total crowd counts in a large area, with high density and using existing surveillance camera infrastructure that does not necessarily have to be high resolution.

3.4 Related technologies

A more advanced and evolved form of crowd counting is known as crowd localization. Gao, Gong, and Li (2021) Crowd localization provides more in-depth data for each instance in the crowd, meaning the position (and relative position) of individuals or groups in a crowd. This is useful for analyzing flow estimation or more precise crowd counting. Crowd localization is still a highly discussed and continuously researched topic. For these reasons, we found crowd-counting technologies more relevant to our project.

4 Analysis

This chapter will describe the analysis of the problem statement and define the outline of the system and functional- and non-functional system requirements.

4.1 System specification

Using cameras mounted around a hotspot of a crowd or on a stationary drone, we aim to develop a platform where an AI model receives video footage from these cameras, uses the model to count the crowd, and helps safety experts recognize a dangerous situation. This could be one or more of the following (Raineri 2004):

- Several people moving into the crowd from a specific direction create a dangerous pressure point
- More than a specified amount of people in a marked area resulting in unsafe conditions (ie. >6 people per square meter)
- Omnidirectional or directional movement in the crowd resulting in a dangerous situation
- An area in the crowd suddenly being void of people, perhaps hinting at a mosh pit or an emergency

These are some of the situations this project aims to systematically detect and alert professionals by providing meaningful feedback.

The system would consist of the following:

1. A backend that receives the data from the cameras, with an implementation of a CNN to handle the video feed according to the bullet list above and exposes this data through an API.
2. A frontend or GUI to display this information in a meaningful way. This could be through a heat map, a numerical estimate of a risk factor, or other visual output.

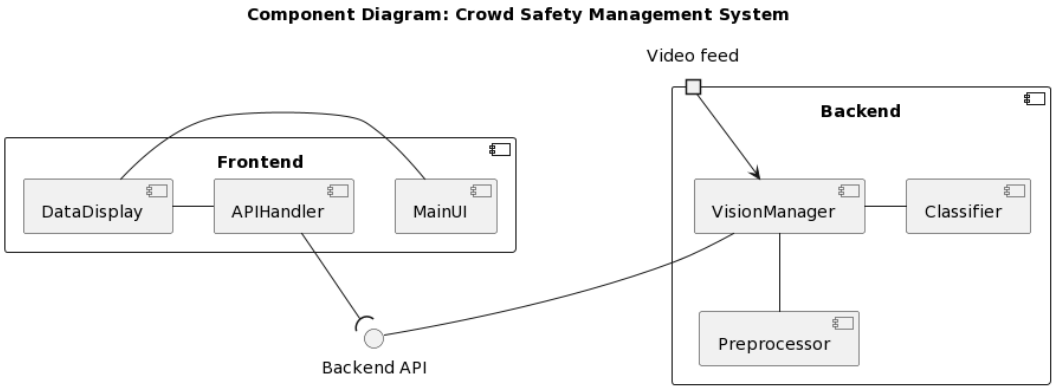


Figure 2: Component diagram of the system

An initial component diagram that maps this described system but is subject to change can be seen in Figure 2.

4.2 Functional Requirements

This section describes the functional requirements for the proposed system along with their "MoSCoW" prioritization ("must have", "should have", "could have", and "will not have"). These are shown in Table 3.

ID	MoSCoW	Requirement
1	M	The system **must** be able to count the number of people in the crowd with the accuracy required by the user base
2	M	The system **must** be able to segment the crowd into virtual sections for further processing
3	M	The system **must** be able to create a heatmap of the crowd density
4	S	The system **should** be able to calculate the physical density of humans per square meter
5	S	The system **should** be able to detect the movement of crowd sections
6	S	The system **should** be able to correct for camera distortion, warp and perspective
7	S	The system **should** be able to detect choke points in the crowd movements
8	S	The system **should** be able to generate a summarising report of the concert with statistics of relevant safety indicators (crowd density, crowd count, risk factors, choke points, and others) after the event
9	C	The system **could** be able to estimate a numerical risk factor based on available factors
10	C	The system **could** be able to generate a live, or delayed, video overlaid user interface
11	W	The system **will not** be able to identify dangerous situations that might call for cautionary actions such as crowd surges, mosh pits, falls, blocked exits, etc.
12	W	The system **will not** be able to use data gathered elsewhere at a concert such as alcohol sales, average crowd age, sound, artists, etc. to give a more precise crowd profile and thus risk factor

Table 3: Table of functional requirements

4.3 Non-functional requirements

This section describes the non-functional requirements for the proposed system along with their "MoSCoW" prioritization ("must have", "should have", "could have", "will not have"). These are shown in table Table 4.

ID	MoSCoW	Requirement
1	M	The system **must** protect the privacy of the personal data
2	S	The system **should** have a user-friendly interface that is easy to manage for both technical and non-technical users
3	S	The system **should** have adequate documentation / technical specification for technical users
4	S	The system **should** have adequate user manuals for non-technical users
5	C	The system **could** have high reliability that is not based on the visual circumstances and environment (e.g. sunlight, stage light, audience flashlights, and other visual effects) or report confidence based on environment
6	C	The system **could** be scalable to simultaneous interoperability between multiple cameras
7	C	The system **could** integrate with existing CCTV software systems at venues

Table 4: Table of non-functional requirements

4.4 Risk analysis

More often than not a project will encounter problems that can hinder the progress or outcome. By being well prepared we can help mitigate some of these risks. Table 5 describes some of the risks we might encounter in this project.

ID	Name	Affects	Description	Mitigation
R01	Code is lost	Project, product	If the code is lost or corrupted	Having a version control system for the codebase and report.
R02	Technology constraints	Product	If the used or available technology is inadequate for the remaining requirements defined in the project	Find alternative technologies or solutions for our requirements before a barrier might be reached. Research the technology and options in depth before implementation. Discuss with our supervisor if alternatives exist.
R03	Personal Conflicts	Product, project	A project-related or personal conflict between the 2 members of the project affecting either the further development or the project as a whole.	Keeping a friendly and open mindset in the work will take us far. Depending on the nature of a potential conflict we would either discuss and solve it outside of work or discuss it with our supervisor.

ID	Name	Affects	Description	Mitigation
R04	Unusable data	Project, Product	If the data provided by the drone or from Event Safety turns out to be unusable due to one or more factors: Resolution, perspective, distortion etc.	Making sure that provided data is on par regarding the requirements of our project. Some data might be fixed by using mathematical algorithms but in general, a proper standard for data is preferred. Can be replaced by data found online, but will not have the same effect.
R05	Company collaboration ends	Project	If Event Safety or the group decides to end the collaboration around the project early.	Making sure that we listen to and adapt to feedback from Event Safety to keep our good relationship with them.

Table 5: Table of risks and their impact

4.5 Risk assessment

To fully assess the risks described above it is vital that the probability and severity of each risk is assessed. This is done by using the following formula:

$$Effect = Probabilty * Severity$$

The probability and severity are given each given a score between 1 and 5. 1 being a low probability/severity and 5 being a very high probability/severity. The effect is then calculated based on these numbers. This can be seen in Table 6.

ID	Probablity	Severity	Effect	Notes
R01	1	3	3	The severity is based on the amount of code lost.
R02	2	2	4	Compute power limitations are the primary factor here. However, we do not see the need for more than what accessible computing resources can provide.
R03	1	1	1	Given the nature of previous work in semester projects and friendship this is highly unlikely.
R04	4	2	8	The severity depends on how bad the data is and how well we can adapt to it.

ID	Probablity	Severity	Effect	Notes
R05	1	5	5	Severity would depend on when in the process Event Safety would cancel our collaboration.

Table 6: Table of risk assessment

Based on the table above, we can deduce that the major risk we should be aware of is R04. How we handle the possibility of unusable data is crucial to the success of this project. While other risks also have high effects, their probability is too low to be considered threatening to the project. However, having measurements in place to counteract every risk is still important.

5 Design

This chapter is going to describe the design choices made for the system as a whole regarding the analysis of the problem in general and requirement specification. This chapter will talk about approaches to crowd counting and how crowd counting differs from object detection. Following this, the chapter talks about the selection process for a cloud computing service and why we decided to go with SASNet as the model for this project

5.1 About the choice of crowd counting

Crowd counting is a field of computer vision research that has developed quickly in recent years. It is a technique that aims to count the instances of any object in an image, no matter the context or density. (Song et al. 2021)

One of the original ideas for solving this problem was to use Meta's SAM model. During the preliminary research sessions, it was discovered that SAM was not adequate in counting people but better suited for object detection. The following citation from Ma, Hong, and Shangguan (2023) highlights the problems well:

"Although the Segment Anything model (SAM) has shown impressive performance in many scenarios, it currently lags behind state-of-the-art few-shot counting methods, especially for small and congested objects. We believe that this is due to two main reasons. Firstly, SAM tends to segment congested objects of the same category with a single mask. Secondly, SAM is trained with masks that lack semantic class annotations, which could hinder its ability to differentiate between different objects. Nevertheless, further exploration of adapting SAM to the object counting task is still worth studying." (Ma, Hong, and Shangguan 2023)

SAM and other object detection methods (sometimes described as "crowd counting by detection") are usually trained on one specific type of object by extracting certain visual features. The benefits of using crowd counting rather than object detection are:

- Crowd counting is less intensive on computing resources.
- Crowd counting performs better in crowded scenes with varying scales of objects and overlapping objects.
- Crowd counting is more robust in environments with changes in light, without the need for specific training data.
- Crowd counting is not as exposed to the risk of overfitting the model. (Li et al. 2021)
- Crowd counting does not single out individuals, but looks at the crowd holistically. (Chan, Liang, and Vasconcelos 2008)

For these reasons, crowd counting is a method that can be used for many domains including traffic and parking analysis, pedestrian crowds, corn and crop counting, and much more. This project will use crowd-counting methods to analyse, count and estimate the density of concert crowds.

(Li et al. 2021) says the current leading approach to crowd counting is the use of a regression-based method, especially in large crowds where detection might not suffice. This paper, contrary to our definition of crowd counting, includes object detection as a method of crowd counting. The main challenge with regression-based methods for crowd counting is to differentiate between the large-scale variations in the countable human heads. (Li et al. 2021) discusses several categories of approaches for regression-based models to solve scale variation problems, one of which is multi-scale fusion. This approach is the category that

this paper proposes for further research, as it shows promising results in the balance between performance and accuracy.

For the reasons outlined here from O'Shea and Nash (2015) and Li et al. (2021), this project will focus on the use of CNN and regression-based models for crowd counting.

One of the useful datasets for crowd counting, "ShanghaiTech" comes from the paper Zhang et al. (2016). In this paper, there are defined two datasets, "part a" and "part b". For the CSMS data, "ShanghaiTech Part A" is a more representative dataset to use, since it contains more densely populated crowds from a bird's eye view. The dataset is labeled with one dot representing roughly the middle of a person's head. This is the approach outlined in one of the founding papers of crowd counting: Lempitsky and Zisserman (2010). In Lempitsky and Zisserman (2010), it is explained how their approach to crowd counting, which has become the primary method in the field, is to create a density function F as a function of the pixels in an image I. This density function is analogous to the physical idea of density, however not a direct representation of the same concept, as physical density is based on physical area. Integrating F over the entire image I will return the number of people in I. (Lempitsky and Zisserman 2010) Each object in the image I should be represented by a normalized 2D Gaussian kernel with the mean at the person's head.

One disadvantage of this approach is that the kernel for objects close to the edge of the image will not be counted as a whole object when summed. It is not assumed that this will pose a significant source of error for the CSMS. The downside of using the crowd counting approach is that it will be difficult to track the movement of members of the crowd, rather than using a detection-based counting method. Regression-based crowd counting might also perform worse for very small crowds as it relies on overall patterns in crowds. Despite this, and because this can be seen as an advantage from a privacy point of view (see Section 5.2.1), the choice was made together with our collaborators to use the regression-based method.

5.2 Design considerations regarding security and privacy

Being citizens of and using the data of people from an EU country, we have to abide by the rules of GDPR. This means that security and privacy around the data that we have acquired are crucial. Practically, this means that we have a responsibility to make sure that we only upload the images and videos we need to run the model on when they need to be there and remove them when not needed. In a perfect world, we would run our program locally to limit the uploading and removal of sensitive data. It is of course also necessary to consider the business and ethical considerations of how using CCTV and computer vision more extensively can affect some peoples' sense of privacy when attending a festival.

5.2.1 Facial recognition vs. Crowd Counting

When reading through this project, one might assume that privacy around facial recognition would be an issue. However, as this is related to heatmaps, crowd counting and crowd density of people, not facial recognition, the identity of people in our footage is not a major concern and thus was not featured in our risk management section. The CSMS anonymizes the output data and does not track individuals through time, which could be a privacy concern.

5.2.2 Selecting a cloud computing service

Initially, the backend of this project was run on a paid but still resource-limited version of Google Collab - a Jupyter Notebook collaboration space with access to cloud computing CPU and GPUs. However, it was decided to research possible alternatives for easier scalability and maintainability. When talking about renting cloud computing GPU, the main factor is cost,

as with enough money you can rent almost unlimited cloud processing power. However, for our project the amount of processing power available was not the primary concern - data security was. This brought our attention away from international services like Google Cloud/vertex and Amazon AWS to a local service: UCloud. UCloud is a capable cloud computing service hosted locally on SDU. Through our supervisor, we took action and applied for resources for this project which were quickly granted. This allowed us to move our initial backend to UCloud, which was a major progression for the following reasons:

1. UCloud is "free" for students and approved projects. You apply for a set amount of currency to use on the platform.

2. UCloud processing power is only limited by the amount of currency available to us on the platform, unlike Google Collab which was limited based on current global usage.

As mentioned, data handling and data privacy are very important for this project. With this in mind, we decided to research the policies of UCloud regarding data handling before implementing our software on the platform. According to SDU Cloud (2020), UCloud is ISO 27001 certified. This means that the platform lives up to an international standard for information security management systems (International Organization for Standardization 2013). Furthermore, according to SDU Cloud (2020), UCloud is hosted locally on SDU and not by a 3rd party operator. Using this information, we can conclude that UCloud as a platform is safe to use for our project and regulated to relevant standards.

5.3 Choosing a technology for Crowd Counting

There are many open-source implementations of crowd-counting models, many of which can be seen in the Awesome Crowd Counting repository by Gong (2023) along with research papers and benchmarks. After careful consideration, the choice of using SASNet was based on four different factors:

1. SASNets fits our need to solve the problem of scale variation in the video material. It was not possible to obtain video material from an approximate orthographic perspective. However, future endeavors for this project might include cameras like those from one of our collaborators PhaseOne who produces approximate orthographic perspectives using high pixel density cameras from high altitudes. SASNet, however, fulfills the requirement to produce relatively high accuracy prediction on an image with perspective for now.

2. SASNet has some of the lowest absolute errors on the ShanghaiTech testing data, according to Gong (2023) which compares at least 45 different crowd-counting models and methods.

3. SASNet is open source and licensed under the permissive Apache 2.0 license, which allows us to use it legally and for free in our project. SASNet also publishes the pre-trained model weights (trained on ShanghaiTech A and B), which means we can save many computing- and time resources instead of training the model ourselves.

4. SASNet publishes the model weights from the training on either ShanghaiTech part A or part B. For this project, it is deemed more useful to use the images from part A. While the labeled dataset is smaller in part A than in part B, it more closely represents the data (i.e. amount of people, density and perspectives) that will be used in this project.

5.4 Backend behavior

Figure 3 illustrates the flow of the program through an activity diagram. The diagram was created to obtain a deeper understanding of the general flow and processes the program goes through during its runtime. The diagram is split into 2 “swimlanes”, one lane for our CSMS system and one lane for the implementation of the SASNet model. In broad terms, this can also be specified as a CPU and a GPU lane where our CSMS system handles CPU processing and SASNet focuses on GPU processing.

As depicted in the diagram, sections of the activities are divided into partitions to visualize a rough partitioning into classes and for better readability throughout the diagram. The input is specified as the path to the pre-trained model and the path to the video for it to run on.

Below is a short run-through of the different partitions.

1. System initialization and input validation.

The CSMS is initialized and the inputs for the model and video path are validated

2. SASNet model setup and construction.

The SASNet model is loaded together with the VGG16 model which it is based on. This sets the foundation for subsequent image processing.

3. Pre-processing of input data.

Pre-processing involves creating a dataset from the video in pyTorch and in turn creating a dataloader from this dataset.

4. Image processing loop.

The images in the dataloader are iterated through and forwarded to the SASNet model until the dataloader is empty. Each image is processed through the different layers to prepare them for post-processing

5. Post-processing and result generation.

Following the image processing, the CSMS takes over again. Here density information is computed, and perspective warping is applied to in turn generate the correct heatmap for further analysis.

6. Output generation and reporting.

All of the outputted data is saved to a file that can be used in the frontend to form a timelapse and a report, summarising the observations.

5.5 Design considerations regarding user experience

For many purposes, a video with a heatmap could be sufficient. However, to make it possible to satisfy requirements such as functional requirements 2 and 8 in Table 3 as well as non-functional requirement 2 in Table 4, it was decided to have a user interface. Ideally, by using a frontend with an API integration to the backend. Alternatively, a frontend where one may upload the data output from the backend themselves, to analyze the data further, is described as a “reporting tool” in functional requirement 8. The frontend should also

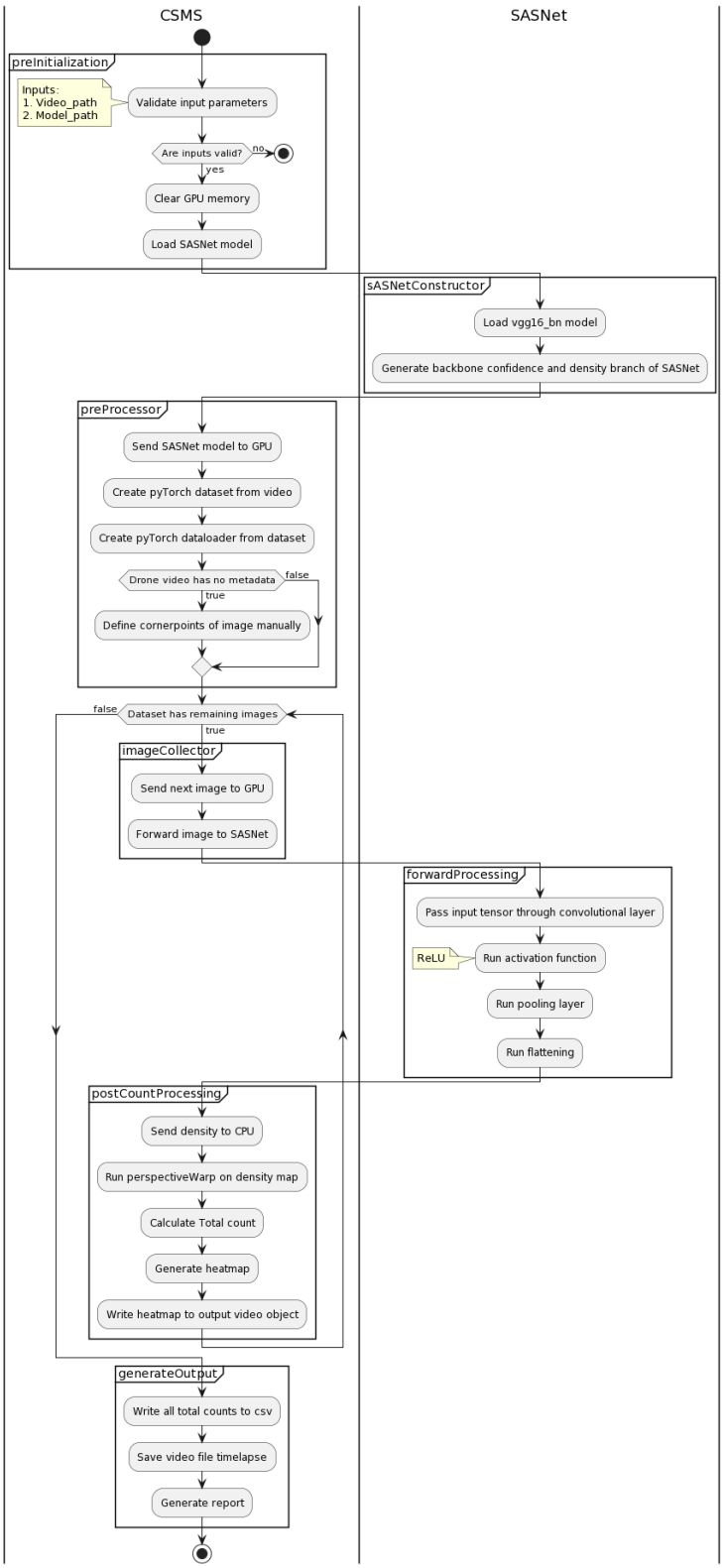


Figure 3: Activity diagram depicting flow in the solution

have a way of changing the color scheme of the heatmap for users who have other color preferences or color blindness.

It is also important to remember that the data produced by the backend is a means of achieving and solving the overall problem statement. Keeping the user in mind, they might experience information overload when looking at a large heatmap with many colors. That is why the backend is going to downsample the heatmap data to the ratio of 1 pixel to 1 square meter. This can be upscaled in the front again to improve the image quality and size, but keeping the ratio of each block on the image corresponding to one square meter. This makes it easier for the users to quickly correlate the information with their preexisting knowledge of service levels (Dahl 2023). Service levels are usually described in people per square meter. Limiting the heatmap color scale to cap out at >5 people per square meter as this is when the density could become worrisome.

5.6 The Cyber-Physical System

Our system could in a cyber-physical aspect integrate the automated sensors (cameras) within a closed network IoT framework to monitor the crowd density. The sensors collect data in real-time, which is then processed and analyzed centrally.

Despite the automated data collection, decision-making remains a human-driven process, with safety experts interpreting the sensor outputs. The interface for this system is user-friendly described in Section 5.5, and could offer real-time visualizations of crowd dynamics. Future enhancements may explore more automated decision-making capabilities. See Section 11.

It is necessary to implement an IoT framework to implement non-functional requirements 6 and 7 in Section 4.3. It would also be necessary in order to implement functional requirement 12 in Section 4.2. However, this requirement is not part of the project scope as it is a "will not" in the MoSCoW prioritization.

6 Implementation

This chapter is going to describe the implementation of as many designed elements as possible within the timeframe, following the prioritization in Section 4 and the design choices made in Section 5.

6.1 Data Collection and Preparation

To implement this project it was necessary to gather useful evaluation data. For this project, that is video security footage and drone footage. Our collaborative partner Event Safety was very helpful in letting us gather this data at their 2 festivals, "Smukfest" and "Grøn". This was done 1-2 months before the project started since this is when the festivals were held. This means that we gathered the data before completing the research on crowd counting and computer vision. It did pose a challenge to collect the data before knowing exactly what type of data was needed. For these reasons, we collected many types of data, including:

1. Drone footage from "Grøn" at an altitude of 25-50 meters from the entrance
2. Drone footage from "Grøn" at an altitude of 25-50 meters of the concerts and end of concerts
3. GoPro footage from "Grøn" at an altitude of 2-5 meters from the entrance
4. Security camera footage from "Smukfest at an altitude of approximately 10-15 meters from the center of the stage.
5. Security camera footage from "Smukfest" at various heights from bars and paths

What turned out to be most useful was option number 2 and option number 4. Having a perspective from a higher altitude means that more people are clearly visible and also minimizes distortion when warping for perspective correction. Having an even higher resolution from a higher altitude could also increase the accuracy of the predictions. The resolution of the drone footage was 1080p. Having a higher resolution could limit the video compression artifacts. The camera options were discussed with our secondary partner PhaseOne. They have experience with ultra-high-resolution top-down drone footage. This could be an option for future endeavors, which will be discussed in Section 11 of this report.

While we, with the help of a certified drone pilot, collected the footage from "Grøn" ourselves, the security footage from "Smukfest" was collected by Event Safety. For this, we created a technical specification with our requirements for the footage at the time. This can be read in Section 12.

6.2 Backend implementation

This section will describe the implementation of the backend. The backend is where the majority of the expensive computing is going to be performed, as well as the use of SASNet and expensive computer vision functions.

To create an efficient, manageable and scaleable backend, an object-oriented design in Python was chosen. In this section, we will look at the responsibilities of each class and how they interact to fulfill the requirements. A class diagram of the backend is shown below in Figure 4.

In this section, we will refer back to Section 4.2 to show the implementation of a few requirements.

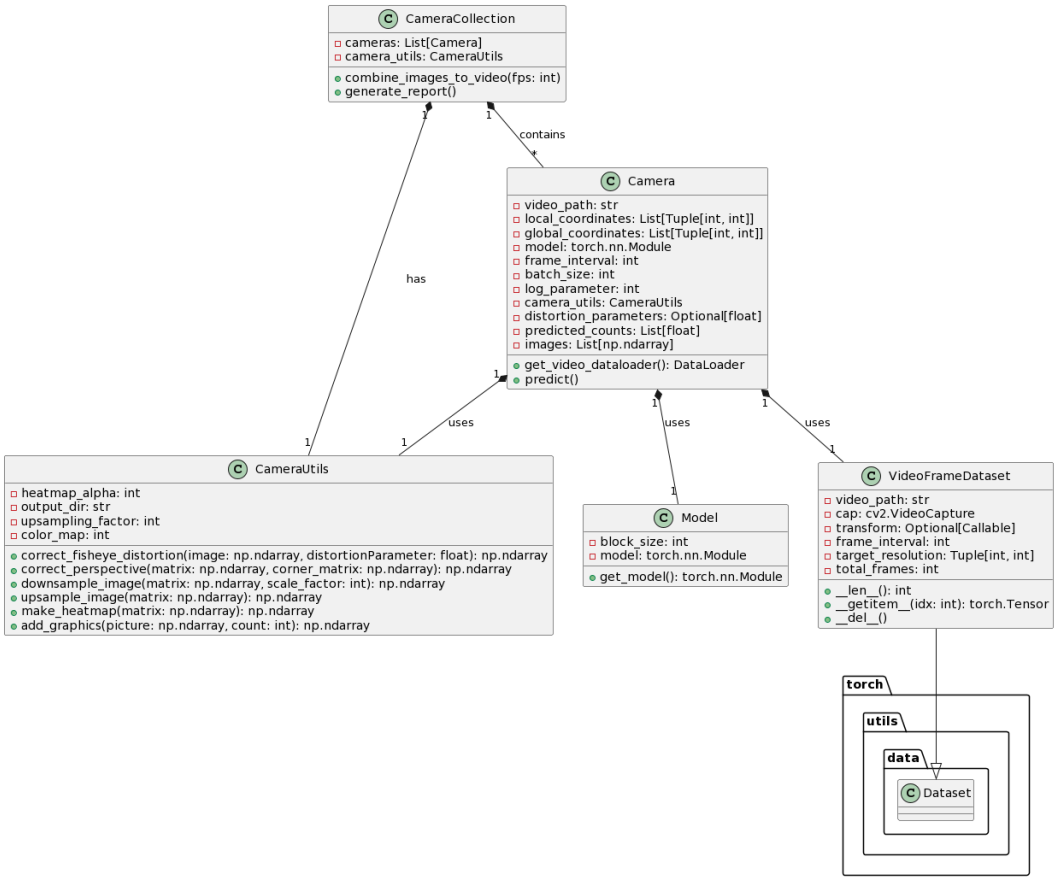


Figure 4: Class diagram depicting the structure of the backend

6.2.1 Backend class structure

Referring to Figure 4, we see that the backend mainly consists of 5 classes:

1. VideoFrameDataset
2. ModelWrapper
3. Camera
4. CameraUtils
5. CameraCollection

The system is built on the OOP paradigm for easier scalability to additional cameras in the future. The responsibilities of the 5 main classes are described in the following:

- VideoFrameDataset

VideoFrameDataset takes the video input and converts it to the type of Dataset. This allows us to segment the video based on a frame interval for easier manipulation later on. Here, the size of the video is also set to a specific resolution based on passed parameters and resized using the cv2.resize() function. The class returns the frames of the video in a Dataset.

- ModelWrapper

ModelWrapper loads the model from the SASNet repository into a Model object with relevant parameters. The class contains a get_model() method for fetching the model in the other classes.

- Camera

The Camera class contains the main predict() method. This method is explained in Section 6.2.5.1. Beyond the predict() method, the class gets the Dataset from earlier and passes it to a Dataloader which the predict method fetches. When the predict() method has run, it returns a heatmap for processing in the frontend.

- CameraUtils

The CameraUtils class is a large collection of methods used in the Camera and CameraUtils classes. These methods include, but are not limited to: correct_fisheye_distortion(), correct_perspective(), downsample_image(), upsample_image() and make_heatmap(). All of these methods are used in the predict() method in the camera class and aid in manipulating the frames of the video for better and more accurate crowd counting.

- CameraCollection

As the main goal of this project is to help improve crowd safety management, the implementation of software that can handle multiple cameras at once is important. The CameraCollection class does just that. The other main responsibility of this class is the conversion from frames in a Dataset back to a video timelapse. The class takes all of the frames and combines them into a video with a set amount of frames per second(FPS) and then saves the video to the .avi format.

6.2.2 Environment setup

Setting up an environment for machine learning in Python is not trivial. It is a requirement to have access to a NVIDIA GPU to take advantage of the CUDA toolkit in PyTorch. This opens up GPU acceleration which enables a performance boost rather than running the CNN models on the CPU. Furthermore, the SASNet open-source project was built for Python 3.6.8 and PyTorch >= 1.5.0 (Song et al. 2021). All of this has to be taken into account when setting up the environment. We created a bash script which can be seen in Section 12 with the following responsibilities to initiate the environment:

- Initiating and cloning Git submodules
- Installing necessary Ubuntu packages
- Installing necessary Python installation
- Installing Python package manager (PIP)
- Installing all Python package dependencies

6.2.3 UCloud service

SDU provides a cloud computing service for faculty members through eScience. Our supervisor applied through this site for a cloud computing resource with 50 GPU hours. The project was granted 100 GB storage and the "u1-gpu @ DeiC Interactive HPC (SDU)" server. In UCloud we used the application "Coder CUDA" to run the project. The bash script from Section 6.2.2 is run to initialize the environment.

6.2.4 SASNet Model Adaptation

The SASNet model is loaded using pre-trained weights and biases for the underlying vgg16_bn model trained on ImageNet (Song et al. 2021). This allows SASNet to more efficiently identify image features with low memory usage. Furthermore, the SASNet weights and bias fine-tunings are loaded into the model which has been trained on the ShanghaiTech Part A dataset.

6.2.5 Implementation of requirements

This section about the backend implementation will take a closer look at how functional requirements 1 and 4 were implemented.

6.2.5.1 Implementing crowd counting

This section describes the implementation of functional requirement 1: "*The system must be able to count the number of people in the crowd with the precision required by the user base.*"

To count the people in a crowd, the SASNet crowd counting model mentioned in Section 5.3 was implemented in the backend. This is done in the model_wrapper.py class. The model_wrapper class has a get_model() method that returns the model for use in the camera class.

Before we can use the model on our data, it needs to be preprocessed. As the SASNet model is made for singular images, converting the input videos to an image-based format is needed. This is done using the Python torch.utils.dataset class. This iterable dataset allows for loading a frame interval from a video into an array of images using cv2. Using cv2, we are also able to resize the video to a specified resolution for easier handling. Once a specified frame interval, resolution and other parameters for the video are defined, it is ready for processing and gets loaded by the camera class.

```
def predict(self) -> None:
torch.cuda.empty_cache()
gc.collect()

dataloader = self.get_video_dataloader()

for img in dataloader:
  img = img.cuda()

  with torch.no_grad():
    self.model.eval()
    pred_map = self.model(img)
  pred_map = pred_map.data.cpu().numpy()

  for i_img in range(pred_map.shape[0]):
    pred_cnt = np.sum(pred_map[i_img]) / self.log_parameter
    self.predicted_counts.append(pred_cnt)

    grayscale_map = pred_map[i_img][0] # Extract the first channel (grayscale) from the densit

    if self.distortion_parameters and len(self.distortion_parameters) > 0:
      undistorted_map = self.camera_utils.correct_fisheye_distortion(grayscale_map, self.disto
    else:
      undistorted_map = grayscale_map

    corrected_map = self.camera_utils.correct_perspective(undistorted_map, self.local_coordina

    width_after = self.global_coordinates[3][0] - self.global_coordinates[0][0]
    scale_factor = int(corrected_map.shape[1] / width_after)

    downsampled_map = self.camera_utils.downsample_image(corrected_map, scale_factor)

    upsampled_map = self.camera_utils.upsample_image(downsampled_map)

    heatmap = self.camera_utils.make_heatmap(upsampled_map)

    self.images.append(heatmap)
```

After loading both the model and dataset, the Camera.predict() method seen above is called. The predict method first clears the garbage collector and empties cached GPU memory. Each image in the dataset is then iterated over and assigned to PyTorch.Cuda, sending it to be processed by the GPU. The model now infers the image and the resulting density map is assigned to the pred_map variable.

After assigning the density maps several post-processing steps happen. First, the predicted count is calculated by the SASnet model. How this works is explained in simple terms in Section 5.1.

Secondly, the method checks if parameters for fisheye correction are passed. If true, the program applies fisheye correction from cv2 and continues, otherwise, this part is skipped.

Third, the first color channel (grayscale) is extracted for further processing. Following this, perspective correction and aggregate downsampling are performed. Upsampling using "nearest" interpolation is also applied if the upsampling factor is different from 1. For use with the frontend, the upsampling factor should be 1 as the upsampling should be done in the frontend. More on this in Section 6.2.5.2. It is important to note that crowd counting is done before perspective correction, as the model is unable to count accurately on the warped image.

Finally, the corrected map is converted into a heatmap using the make_heatmap() method from CameraUtils if a heatmap color scheme is provided. For use with the frontend this should not be provided as the heatmap will be applied in the frontend. The resulting image is returned to be written to a video file.

In CameraUtils we also find the add_graphics() method, simply adding a small graphic to the image with the predicted count if it is wanted to use the video without the frontend.

6.2.5.2 Implementing the functions to create heatmaps This section describes the implementation of functional requirement 4: "*The system should be able to calculate the physical density of humans pr. area unit*"

Before explaining how functional requirement 4 is implemented, a look at how downsampling and perspective correction work in our system is needed.

When a camera is instantiated in the main class, a set of local and global coordinates are passed with it, as seen in the code snippet below.

```
local_coords = [(797, 293), (287, 653), (1761, 1040), (1734, 411)]
global_coords = [(0, 0), (0, 80), (100, 80), (100, 0)]
```

The local coordinates are taken from the frame of the video, depicting the area we want to analyze and correct. The global coordinates are the real-world distances between points, taken from various land surveys of the different festivals provided by our partner at Event Safety. Using these coordinates as measurements we can correct the perspective of each frame for accurate density heatmaps based on physical distances.

Perspective correction is done using the cv2 method which takes the extracted grayscale map from earlier and the local coordinates from the frame. First, the transformation matrix M is calculated using the original input coordinates and the desired output coordinates. See Figure 5.

After finding the transformation matrix, the operation is performed on the section of the image defined in local_coordinates, using the warpPerspective() method from cv2. The process can be seen in Figure 6.

The code snippet for perspective correction is also found below:

```
def correct_perspective(self, matrix: np.ndarray, corner_matrix: np.ndarray) -> np.ndarray:
 pt_A = corner_matrix[0]
 pt_B = corner_matrix[1]
 pt_C = corner_matrix[2]
 pt_D = corner_matrix[3]

 width_AD = np.sqrt(((pt_A[0] - pt_D[0]) ** 2) +
 ((pt_A[1] - pt_D[1]) ** 2))
```

```
width_BC = np.sqrt(((pt_B[0] - pt_C[0]) ** 2) +
((pt_B[1] - pt_C[1]) ** 2))

maxWidth = max(int(width_AD), int(width_BC))


height_AB = np.sqrt(((pt_A[0] - pt_B[0]) ** 2) +
((pt_A[1] - pt_B[1]) ** 2))

height_CD = np.sqrt(((pt_C[0] - pt_D[0]) ** 2) +
((pt_C[1] - pt_D[1]) ** 2))

maxHeight = max(int(height_AB), int(height_CD))

input_pts = np.float32([pt_A, pt_B, pt_C, pt_D])
output_pts = np.float32([[0, 0],
                         [0, maxHeight - 1],
                         [maxWidth - 1, maxHeight - 1],
                         [maxWidth - 1, 0]])
```

When the perspective of the frame is corrected, the image is then downsampled to reduce information overload and increase the crowd overview by having each "pixel" of the frame correlate to 1 square meter in real life based on the global coordinates. After this, the image is then upsampled again so the data is still readable but not overwhelming.

The code for the downsample_image() method is seen below:

```
def downsample_image(self, matrix: np.ndarray, scale_factor: int) -> np.ndarray:
    # Calculate the dimensions of the downscaled array with padding
 new_height = matrix.shape[0] // scale_factor + (matrix.shape[0] % scale_factor > 0)
 new_width = matrix.shape[1] // scale_factor + (matrix.shape[1] % scale_factor > 0)

 # Initialize the downscaled array with zeros
 downscaled_array = np.zeros((new_height, new_width), dtype=matrix.dtype)

 # Iterate through the downscaled array and calculate the sum of the pixels
 for i in range(new_height):
   for j in range(new_width):
     row_start = i * scale_factor
     row_end = min((i + 1) * scale_factor, matrix.shape[0])
     col_start = j * scale_factor
     col_end = min((j + 1) * scale_factor, matrix.shape[1])
     downscaled_array[i, j] = np.sum(matrix[row_start:row_end, col_start:col_end])

 return downscaled_array
```

This leaves us with the steps condensed into:

1. Correct the perspective of the heatmap based on global coordinates.

2. Downsample the corrected greyscale heatmap to ensure that each pixel in a given frame corresponds to area units - square meters in this case.
3. Upsample the downsampled image for better readability while still maintaining a linear relationship between pixels and area units. (This is only done if the frontend is not used)

After the complete image processing and the steps above are complete, the greyscale heatmap is uploaded to the frontend. Here the pixel sum of the entire image is divided by the amount of pixels to find the density. This is due to the density of people being represented by pixel density as explained in Section 5.1.

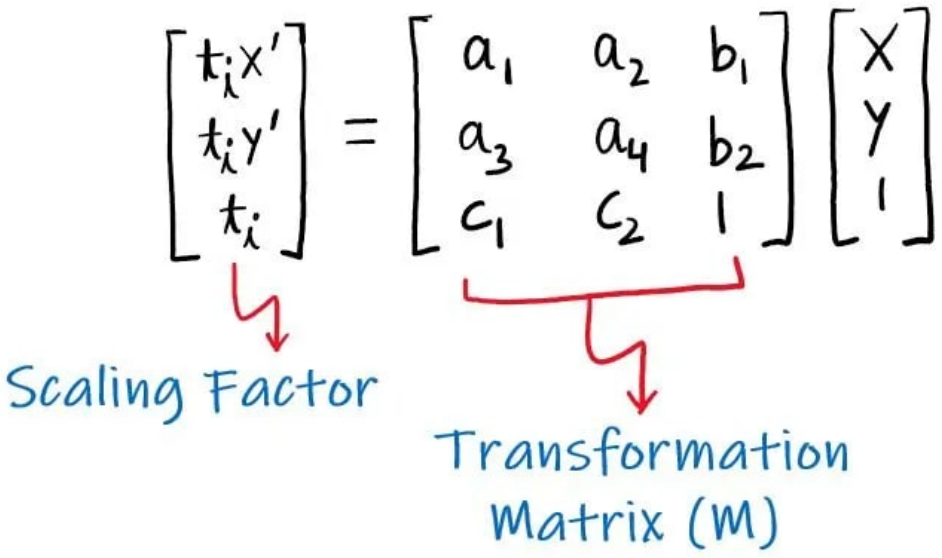


Figure 5: Depiction of transformation matrix and scale factor (From The AI Learner, n.d.)

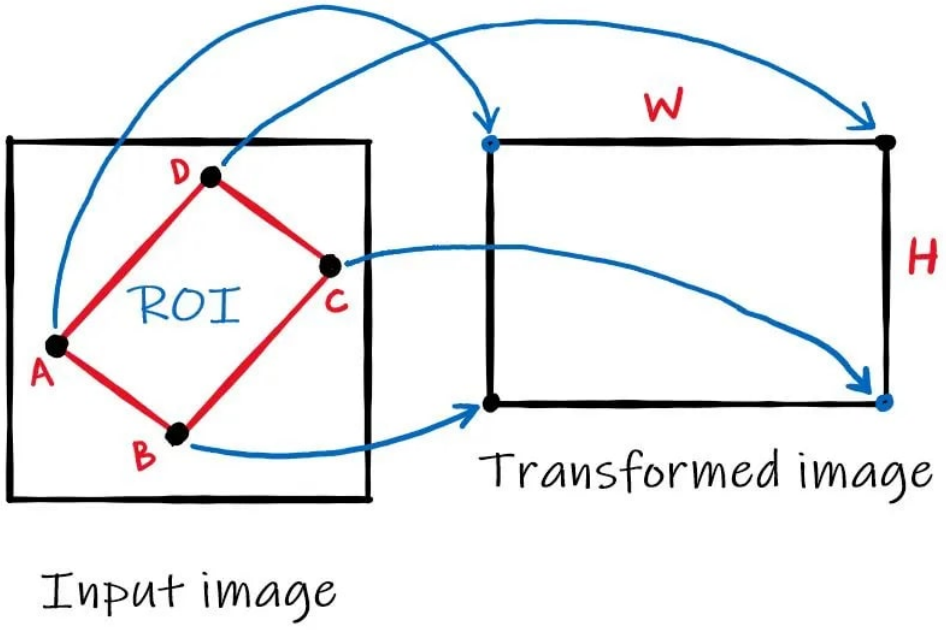


Figure 6: Depiction of perspective correction from The AI Learner (n.d.)

6.3 Frontend

To create a flexible user interface, the output data was stripped of all post-processing artifacts (heatmap, scale, count and other text) as this would now be calculated in the frontend. The frontend is created using Qt for Python to allow for cross-platform compatibility, to have access to video playback components and the Python OpenCV module to allow for preprocessing of the video. There are 3 main widgets in the frontend. The main VideoPlayer widget has buttons to upload the video, clear mask and change the heatmap color scheme. The InfoWidget displays textual data such as the count, count in the selection, density, density in selection and time passed. The FloatingOverlay is an overlay that follows the size and position of the VideoPlayer where the user can draw a polygon to define an area on the video that should be counted, as defined in functional requirement 2 in Table 3.

A simplified version of this structure can be seen in Figure 7 and the full diagram with all methods and attributes can be seen in the appendix. The VideoPlayer calls a preprocess function that adds the selected heatmap and upscales the downsampled video before sending it to the QVideoWidget. A density scale is also created after the video has been uploaded. All of this is done using OpenCV. This is done in accordance with functional requirement 3 in Table 3.

6.4 Integeration

Although a proper API between the frontend and backend was described in the component diagram, Figure 2, it has not been a priority of the project and has thus not been developed. Further development could allow for direct upload to the backend processing unit from the frontend. Instead as of right now, all video content must be uploaded to UCloud for processing which then produces an output video file in the .avi file format that can be read and processed properly by the frontend. Pairing this file with a metadata file containing

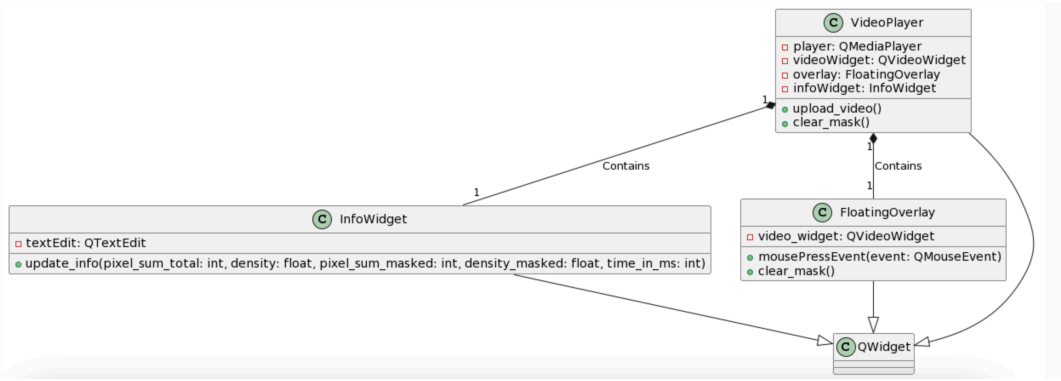


Figure 7: Simplified class diagram depicting the structure of the frontend

the input parameters to the backend such as frame sampling interval could also be useful so these parameters become dynamic in the frontend. The metadata file could also contain total counts and average densities over time, so these expensive operations do not have to be unnecessarily recalculated in the frontend.

6.5 Optimisation and Performance

For two main reasons, it is necessary to limit the compute resource usage:

1. This project is using limited GPU and compute resources provided by UCloud. Computer vision and CNN's can be very GPU intensive, so it is in the project's interest to limit the GPU memory consumption as this was found to be the largest resource bottleneck.

2. For the CSMS to be scalable to (near) real-time evaluation performance, it makes sense to limit the use of computing power as much as possible.

According to Contributors (2023), disabling gradient calculation is useful for lowering memory usage when using a model for inference. This fits the use case and greatly reduces memory usage. This was done in the following way:

```
img = img.cuda()
with torch.no_grad():
  model.eval()
  pred_map = model(img)
pred_map = pred_map.data.cpu().numpy()
```

Here the image tensor is first sent to the NVIDIA GPU using img.cuda(). Gradient calculations are then disabled for the inference, and the model is set to eval mode. These two reduce the GPU memory usage. Finally, after the inference is run and the predicition_map is calculated, the tensor data is sent to the CPU and the tensor is converted to a NumPy matrix (an image). This way limits the time that the tensor spends in the GPU greatly, as well as reducing memory usage.

7 Validation and Verification

Since this project is a proof of concept, automated unit tests and integration tests are not as favored as user tests and similar. Because of the innovative nature of the product, validation of the idea is more important than verification. Testing this system relies on making sure that it is useful and understandable to those who need it. To accomplish these tests, we decided to perform both manual count comparisons and get the opinions of Event Safety security personnel and professionals about our project and its viability.

7.1 Accuracy of Crowd Count

To test the accuracy of the model, a manual test was conducted on a small area of the video.

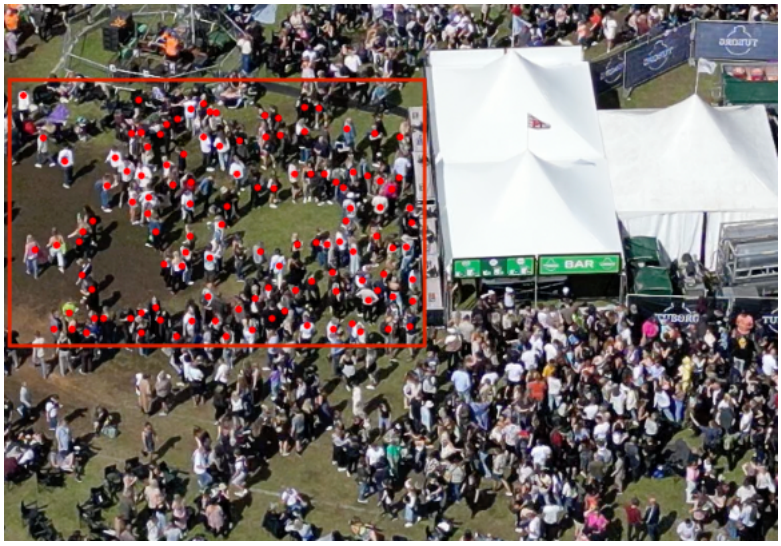


Figure 8: Small section of bar after concert - Manual count

In the image seen in Figure 8, a section of the queue to a bar towards the end of a concert. A rough manual count of the visible people was conducted. This resulted in a count of around 120 people

Now, the same area is highlighted in the frontend to count the people using the model. This can be seen in Figure 9.

While this test shows that the model is precise in optimal circumstances, there are of course limitations. The model can only count as well as a human would be able to. This means that under circumstances where lighting, stage artifacts (Confetti, fireworks etc.) or bad image quality is present, the model is unable to count precisely. More on this in Section 9

7.2 Runtime and GPU usage

While GPU usage is something already discussed in implementation, runtime and the optimization of this are still crucial to test. While we originally wanted this project to be able to run in real-time, we quickly realized that this was close to impossible with our current setup and implementation. When running the model on a video, we originally ran it on a 3-minute clip with an image every 30 frames being processed. This resulted in a 180-frame video and a runtime of the program of around 15 minutes. In the future, if we want to run

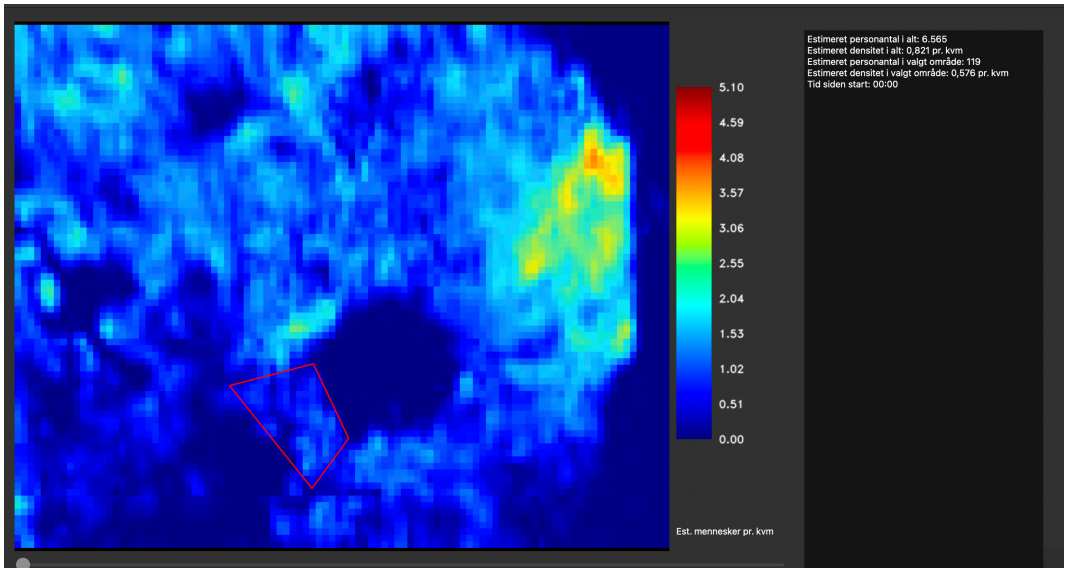


Figure 9: Small section of bar after concert - Model count

this live, additional resources or optimization is required. Alternatively, the frame sampling interval could be adjusted to match the speed of the inference process, which would result in a very low frame rate, but likely still useful. More about this in Section 11.

7.3 Event Safety presentation

Based on the analysis, design and development of the crowd management system we were invited to present our project to more people from Event Safety. Research was conducted into how a proper focus group is structured and managed properly, to give the best possible feedback and evaluation on the project.

7.3.1 Focus group

A focus group's purpose is described as: "As a summative evaluation, focus groups can be used to assess the acceptance of a new campaign or gauge customer satisfaction levels" (Jenn and Visocky O'Grady 2017). This fits the needs and possibilities for this evaluation. Based on research from this book we arrived at the following conclusion for our focus group:

- A group of 6-9 people is the preferred group size
- A group of similarly experienced people is needed to encourage a proper discussion
- If possible, have more groups with different people

For the presentation, a group of 8 employees from Event Safety attended the presentation with questions and feedback for us along the way. The employees all had major experience with crowd management and event planning. Most of them had heard of our project beforehand but had no further technical insight. The presentation consisted of a short introduction to the background, context and process, a live demonstration of the CSMS containing real examples of situations and concerts they know about, followed by a questionnaire that gives quantitative and qualitative data on the evaluation.

The focus group session started with an exercise, asking them to estimate the amount of people in a large crowd section. They were only given the meta information such as the time

of day, place and size of the area. This was intended to help us validate the idea. Their range estimates varied from 2.000 to 12.000 in a section with 6.000 crowd members. This is often the current process when making density and crowd count estimates, although sometimes helped in real life by other data such as turnstile counts and tickets sold. This validates the system because of the large uncertainty in their estimates. Safety guards could be working against each other's interests if one believes there are 2.000 people in an area, while another believes there are 12.000 people.

7.3.2 Focus Group Questionaire

Following the focus group, a questionnaire is conducted to gain further insight. Normally, a questionnaire is used to gather information from a large group. For our presentation, we still felt a questionnaire was relevant in order for us to be able to document their answers in a written format.

The questionnaire was conducted using Mentimeter and consisted of scales, open-ended questions and rankings. This section will highlight some of the answers the attendees provided. All the questions and their results (in Danish) can be found in Section 12.

1: To which degree can you see the usefulness in this system for live events and for post-event evaluations?

In question 1 the participants are asked to verify the need for the product and the system as a whole. They are asked what they think the usefulness of the system is as a whole for post-event evaluations (4,6 / 5) and as a live system (4,4 / 5). This verifies that there is a need for the product and that it has been a correct prioritization to focus on a post-event evaluation tool while still having a use case in making the process live in the future.

2: Non-functional system properties

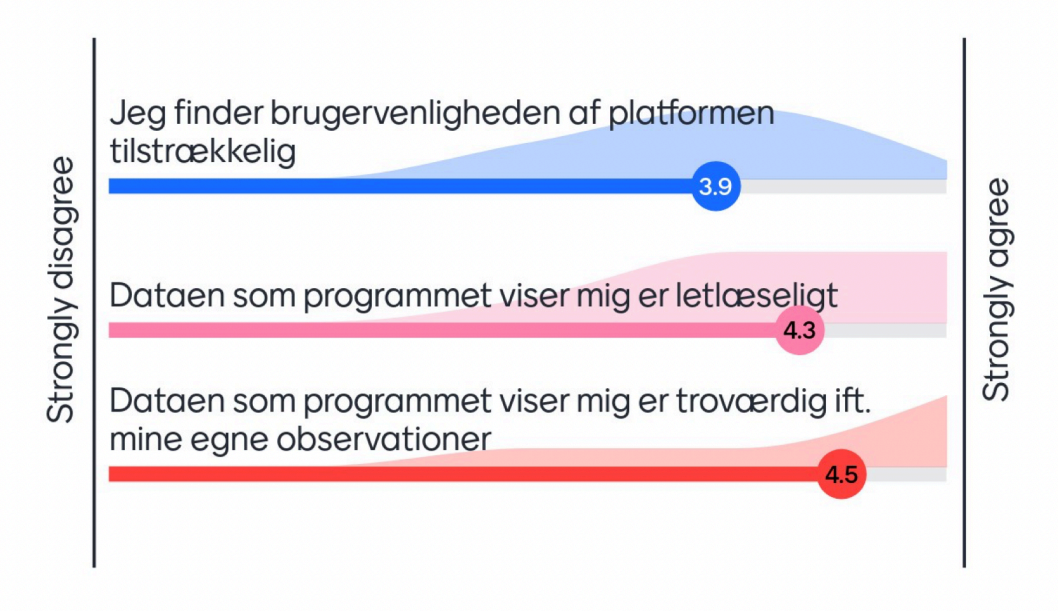


Figure 10: Questionnaire question 2

Question 2 had the attendees evaluate 3 statements on a scale of 1 to 5 and "strongly disagree" to "strongly agree". 1 being disagree and 5 being agree.

The 3 statements were as follows:

1. I find the usability of the platform adequately simple.
2. The data the program visualizes is easy to read and understand.
3. The data the program provides is credible compared to my own observations.

The first statement was evaluated to a score of 3,9, showing that while the platform in general seems intuitive and fulfills non-functional requirement 2, there is room for visual improvements.

Statement 2 further validates the implementation of non-functional requirement #2 with a score of 4,3.

Statement 3 shows that the professional observations from our attendees align with the output of our program, showing that functional requirement 1 is fulfilled.

An attendee also mentioned that the heatmap provided by the program gives more value than a precise count of individuals in a crowd. This comment was unfortunately not documented.

3: Accuracy and uncertainty of the system

Figure 11: Questionnaire question 3

Question 3 had the attendees evaluate 2 statements from 0 to 100% based on their preferred uncertainty of the data provided.

1. What percentage of uncertainty would you prefer?
2. How high of an uncertainty would you be willing to accept?

Statement 1 shows a score of 2,2, meaning an uncertainty of around 22% would be ideal for the crowd-counting values provided by the program. This is useful to evaluate functional requirement 1.

Statement 2 shows a score of 3,7, meaning an uncertainty of around 37% would still be useful. This further ties into the comment made to question 2, that the precision of the counts is not as important as the visualized density maps - though these do rely on each other.

It is also important to note that the required uncertainty percentages vary depending on the total size of the crowd.

7.4 Evaluation of functional requirements

Based on the results and evaluation of our presentation for Event Safety personnel, we can say the following about our functional requirements:

1. The system **must** be able to count the number of people in the crowd with the precision required by the user base

The system can use the model to count crowds of people where a human would be able to. When lighting conditions or materials from a concert (bright lights, confetti, fireworks etc.) are present, crowd counting precision is dramatically reduced. Based on answers in question 3 and previous conversations with Event Safety, we learned that a higher precision is not as desirable as a proper heatmap. This meant that focus was put on perspective warping and generating a proper heatmap of density in favor of *very* precise crowd counting.

2. The system **must** be able to segment the crowd into virtual sections for further processing.

When the processed video is uploaded to our frontend, the user can select a polygon for further processing. The program provides a count and density value for the specified area.

3. The system **must** be able to create a heatmap of the crowd density.

The system creates a heatmap of the analyzed video to visualize crowd density. The user can select the color scheme of the heatmap themselves before uploading a video.

4. The system **should** be able to calculate the physical density of humans per area unit.

The system can calculate and display the physical density based on the sum of pixel values in a given area, as each pixel value corresponds to a certain amount of people. This is dependent on a land survey.

5. The system **should** be able to detect the movement of crowd sections.

The system is not directly able to detect the movement of a crowd. However, the system does create a timelapse of the area in a video which can be used to manually follow certain crowd movements. According to the focus group evaluation, it is a desired feature to be able to follow the flow in and out of a selected area.

6. The system **should** be able to correct for camera distortion, warp, and perspective.

The system can correct for perspective by providing ground measurements and image coordinates to the cv2.warpPerspective() method of the OpenCV library. The system cannot currently correct for fish eye distortion.

7. The system **should** be able to detect choke points in the crowd movements.

   The system is not able to detect chokepoints beyond manual observations of the users.

8. The system **should** be able to generate a summarising report of the concert with statistics of (crowd density, crowd count, risk factors, choke points, and other) relevant safety indicators after the event.

The system is not able to generate a report of every statistic provided here. The report is based on the current frame of the timelapse and shows crowd count and crowd density.

9. The system **could** be able to estimate a numerical risk factor based on available factors.

   The system is not able to estimate a numerical risk factor.

10. The system **could** be able to generate a live, or delayed, video overlaid user interface.

    The system can generate a delayed video user interface based on the input. It is not live.

7.5 Evaluation of non-functional requirements

Likewise, we can say the following about our non-functional requirements

1. The system **must** protect the privacy of personal data.
   The system only handles sensitive when analyzing the surveillance footage. The only way the data is ever close to online is when it is processed on UCloud. However, as the only online part of the process is the usage of UCloud GPUs, the data is never compromised.

2. The system **should** have a user-friendly interface that is easy to manage for both technical and non-technical users.
   The system has a user-friendly interface that is easy to manage for both technical and non-technical users. This is based on the answers provided in Section 7.3.1. With an average score of 3,9, there is still room for improvement.

3. The system **should** have adequate documentation/technical specifications for technical users.
   The backend has a README.md file which instructs technical users on how to run the backend as well as system requirements. This can be seen in the backend/README.md and in Section 12.

4. The system **should** have adequate user manuals for non-technical users.
   The frontend has a user guide with instructions on how to use the frontend CSMS and the videos produced by the technical users in the backend. This can be seen in Section 12.

5. The system **could** have high reliability that is not based on the visual circumstances and environment (e.g. sunlight, stage light, audience flashlights, and other visual effects) or report confidence based on the environment.
   See Section 7.4

6. The system **could** be scalable to simultaneous interoperability between multiple cameras.
   The system can provide output from multiple sources of input at the same time. While

this should also be a functional requirement for it to be fully satisfied, the system has been implemented with this in mind, making it easily scalable to multiple cameras. The focus group evaluation showed that this is one of the most wanted properties of the system.

7. The system **could** integrate with existing CCTV software systems at venues.
   The system is not able to integrate with existing CCTV beyond the footage provided to it.

8 Operation

This section will describe the plan for operation. Due to the seasonal nature of festivals, it has not been possible to test the system in operation at a real festival or concert, nor has it been possible to collect more or improved data. Thus the "operation and maintenance"-phase of the Waterfall model from Sommerville (2011) has not been a priority in the project. The project's purpose was as a proof of concept from the beginning. However, the system has been designed with maintainability in mind

The frontend and backend code bases are maintainable for a few reasons. The Python code has been written using type hints. Even though typing is not supported by the Python runtime "Python Documentation" (2023), it is used by IDEs and linters to enforce good coding practices. The system is also written using an OOP paradigm which makes it easier to understand due to the abstraction of implementation. Writing the code in pair programming also helps maintainability as knowledge is shared and not isolated. When necessary, this knowledge has been documented in README files.

There are no developers in Event Safety. This means that there have not been any considerations for development handover or for second part developers to get involved in the process. For the product to get applied in their day-to-day work it would take building up a small catalogue of processed videos of different examples from different venues. The frontend could be handed over to the users. Users would be employees and volunteers at Event Safety, Smukfest and Grøn Koncert. This would provide significant value based on our focus group research.

9 Discussion

This section will discuss our results including the accuracy of the CSMS and qualitative user feedback from our partners. It will describe how this system could affect the crowd safety management flow. It will also discuss the current system's shortcomings and what can be done in the future to prevent this. Then it will discuss how this system is an improvement from the status quo. It will discuss the privacy and ethical concerns that might be with a system that deals with personal data. Finally, it will also evaluate the process of developing this system and the choice of technologies.

To evaluate the accuracy of the developed system we first look at the quantitative benchmark of SASNet. Here it was stated that SASNet performs better or on par with several other state-of-the-art crowd-counting models. On the ShanghaiTech part A specifically, which the model weights used in the CSMS are trained on, SASNet performs with a mean average error of 53.59 (Song et al. 2021). According to Modolo et al. (2021), the Shanghai Tect Part A has an average of 500 people in each image, meaning the average error percentage is 10.6% on this dataset. This is the dataset that most closely matches our spread and density distributions in the collected data. When doing our testing as described in Section 7.1 we achieve a similar, good accuracy. The required accuracy according to our user research in Section 7.3.1 is that an accuracy of less than 10-20% is ideal, but 20-40% might still be useful. This is well within the bounds of the accuracy of the CSMS.

However, our accuracy is limited by the important factor that our system as a whole aims to calculate the actual crowd count of the physical space and not just the crowd count of the countable crowd members on the video footage. As an example, we would likely achieve a 0% error if the camera was low enough to only count a handful of members, but this is useless in regards to the overall problem statement of improving the crowd overview for security guards. It is uncertain how prevalent this type of uncertainty is in our data, as we do not have this real ground truth. The limiting factors for achieving this goal are:

1. The cameras must have a high point of view so the maximum amount of crowd members are visible.
2. The lighting must be good enough to distinguish members of the crowd.
3. Stage effects (smoke, confetti) must not cover members of the crowd.
4. Video compression artifacts must not be too prominent.
5. The camera must be in focus.

While the collected data for these purposes is a good starting point, it is important to have these factors in mind when collecting data in the future. Some of these factors are difficult to affect because of other circumstances. Number 1 in the list above is limited by drone law regulations or the height of the stage or nearby masts. Number 2 is difficult during night concerts. It could be possible by installing night vision or infrared cameras, but it is uncertain how SASNet would perform with this kind of input data. Number 3 is not possible to affect directly, but camera angles could affect how stage effects affect the input data. It might also be useful to sample frames from moments without many stage effects. Numbers 4 and 5 should be manageable by using proper camera setups.

Based on the section Section 7.4 the CSMS at this point fully satisfies all the "must have" functional requirements and 1 "should have" requirement (requirement 4). It also satisfies most of functional requirements 6 and 8, while functional requirements 5 and 7 are not directly implemented. The "could have" requirements 9 and 10 have not been implemented.

In regards to our evaluation of the non-function requirements in Section 7.5, the "must have" and "should have" requirements are fulfilled. The "could have" requirements are partly fulfilled. Particularly the "could have" requirement 6 regarding "interoperability between multiple cameras" was paid attention during the implementation as this was found to be important for the company. According to our focus group results in Section 7.3.1, the good feedback, on the trustworthiness of the output data and the usefulness of the heatmap, validates the implementation of a large part of the functional and non-functional requirements. The qualitative feedback from the focus group also verifies the need for the problem statement and the usefulness of a system like this in general compared to the status quo, which is manual estimates of crowd count and densities. According to Section 7 The CSMS is both faster and more accurate than manual estimates.

The project's collaborators did not have many ethical concerns when asked. However, it is still our responsibility as the experts, to have these considerations. That is why we researched the data policy of the cloud provider UCloud which satisfied the need for data privacy and responsible use. The ethical considerations were also one of the reasons why we decided to use the regression-based crowd-counting method of SASnet rather than "crowd count by detection". The crowd-count-based methods remove the detection of individuals as each output pixel could be an aggregate of many Gaussian kernels, as described in Section 5.1. The extent of discriminatory biases in the dataset and algorithms such as racial, gender or handicap has not been tested. Using a crowd count algorithm should be less biased than the counting by detection algorithm, making it unlikely that it has a large impact on the final result. However, the choice of crowd counting method implemented some functional requirements, in regards to tracking the movement of crowd members, more difficult to implement which resulted in them not being implemented. A workaround could be implemented while still using SASNet by looking at delta changes in densities over multiple periods, but it is not trivial to infer the average paths of crowd members (functional requirement 5).

Following these considerations and results, "crowd count by detection" and specifically SASnet's way of solving the scale variation has proved to be a reliable, accurate and fast way of crowd counting, while also reducing the potential ethical concerns of a system that creates data of tracked individuals.

The planned timeline of the project was followed and did not prove to be too tight. The timeline could have been slightly improved by also having a way of gathering data after having learned the lessons of the implementation phase, but this was not possible due to the time of year. Some would perhaps argue that the fact that we were able to follow the planned timeline is a sign that more work could have been done, but that was not the case. The prioritiation of the requirements and planning of the phases resulted in a system that we were able to finish within the timeframe while still creating a product that was useful for our target group. The project was completed using a waterfall process rather than a Scrum process. This was done because the later stages in the design and implementation phase depend on the knowledge of the earlier stages. This choice has been a good process for the project since we ended up with a finished and working proof-of-concept product. A Scrum process could have resulted in difficulty in meeting deadlines because of the group size of 2.

10 Conclusion

In conclusion of this project on the Crowd Safety Management System (CSMS) that uses SASNet for crowd counting and density estimation at festival events, we look back at what was achieved and the challenges faced. This is done with regards to the original problem statement from Section 2.3 that is: "Can computer vision software and AI techniques be leveraged to improve crowd overview for security guards, by receiving video feed from large crowds, and ultimately improve crowd safety?"

The main success of the project was incorporating a crowd-counting model, specifically SASNet, effectively for density estimation. This method was very accurate in many crowd situations, which was crucial for the system's usability at the festivals Grøn and Smukfest. We tested the system using data from different festivals and under different densities and circumstances and found success in most common scenarios.

A key part of our work was following GDPR, data privacy and ethical considerations. This was important because we were dealing with sensitive video data. We used the crowd count method to keep individual identities private and instead used aggregate sums of people per square meter, showing our commitment to ethical data handling and maintaining the public image of the festivals to festival guests that they and their sensitive data are safe.

Creating a user-friendly interface was a non-functional requirement. We wanted the system to be easy to use for both technical and non-technical users. The interface we developed helped users easily understand complex data, like heat maps and crowd density figures. It was found from the focus group session that the heat map is very useful in improving crowd overview for security guards at post-event evaluations.

We chose to use regression-based methods for crowd counting to avoid more intrusive methods like tracking individual people. This decision was important to respect individual privacy and align with ethical standards in surveillance technology. The technology also proved to be useful when considering the needed computing power for the alternative.

The project also expectedly faced challenges. Physical environmental factors like poor lighting and stage effects sometimes affected how accurately the system worked. These are common issues at festivals and make maintaining consistent performance difficult.

There were also technical challenges. The system did not fully develop some features that could be interesting for managing crowds, like detecting crowd movements and identifying potential choke points. However, during user testing and discussions, it was no features that were missing for achieving the overall goal and proving the concept of the system. Integrating the system with existing CCTV systems was also a requirement that was not developed because it would not be able to be tested.

Handling and processing a lot of high-resolution video data was challenging. We had to balance the need for powerful processing and storage with the requirement for real-time or near-real-time data processing. This was a tough task given the amount and type of data we were dealing with. The delay showed to be around 4-5 seconds for each frame which is a low enough delay for the use case.

Our focus group evaluation in Section 7.3.1 shows that experts on crowd safety find this system extremely useful as a tool in their work with crowd safety and crowd comfort, successfully proving this technology and system as a proof-of-concept.

Overall, this project has successfully proved the usefulness of AI and computer vision techniques for crowd safety management to our company collaborators. We faced and overcame many challenges and complexities, creating a system that improves not only crowd safety but also crowd comfort at festivals. The experiences and knowledge gained from this project provide a solid base for future developments and improvements in crowd management and safety systems. These will be summarised in section Section 11.

11 Future Perspectives

There are many opportunities for further development of this project in collaboration with the company collaborators, under the assumption that the ethical considerations in Section 9 are kept in mind.

The current implementation is mostly suitable for post-event evaluations. This was decided during the design phase, as this would produce a usable product that was possible to implement. It is, however, also a possibility to adjust the implementation to allow for a live feed of the heatmaps, with as little delay as 4-5 seconds on each frame with the current performance described in Section 7.2. Seeing the heatmaps could open up completely new use cases for the users and possibly change the workflows of the crowd safety managers and security camera operators.

The system's use cases are not limited to festivals and concerts. This can of course be used by any authority or organisation in charge of large gatherings of people. This could be other users like police, defense authorities, emergency management agencies, transportation authorities, sports stadium managers, retail and shopping malls, and theme park managers, for handling events like protests, public national celebrations (eg. sports celebrations, royal birthdays), sports events, special days resulting in high-density gatherings like Black Friday, New Years and Christmas. The only requirement is the ability to collect usable video input data from a high altitude viewpoint, and a group of people with the necessary knowledge of crowd safety willing to use the information provided by the CSMS in a meaningful way to make decisions.

During the design and implementation phase, a multi-camera setup was briefly investigated as a way of improving the output from venues with many smaller corridors and no possibility for aerial top-down views (such as Smukfest). While the collected data was not ideal for this idea, the current implementation is set up to allow for this, by defining multiple cameras and their physical bounds.

It could be an interesting perspective to incorporate more cyber-physical or IoT aspects into this system. Currently, the system consists of a sensor system with video as input that helps crowd safety managers in their decision-making. The system could implement an automated control system for adjusting density imbalances by controlling entrance flow, affecting human behavior by pushing information through an app, stage LED screens or public address systems. The sensor system could also be improved by incorporating more data sources into the algorithms and predicting the crowd behavior, so decisions could be made in due time. This could be done by implementing an IoT system with a network of sensor devices such as data from bar sales, turnstiles, crowd profiles, data from cellphones, etc. and integrating the data into an AI prediction algorithm.

During our discussions with the drone company PhaseOne, they demonstrated some interesting technologies regarding drone and camera technology where this system could be applied. The company provides software and hardware for extremely high-resolution imagery on drones. Using this technology it could be possible to cover a very large geographical area using only one or few drones from a high altitude, making it possible to cover multiple or spread out gatherings at once while requiring almost no setup time. Their technology can also allow for approximate orthographic projections, reducing or completely removing the need for perspective warp. This technology could be very useful for open-air festivals with many stages at once, protests and urban gatherings that are often spread out and high density.

Another interesting idea for further development of this project would be to advance the project from crowd counting to crowd localization. As described in Section 3, crowd localization is a more advanced, proposed form of crowd counting. Advancing an implementation

of this would allow us to further increase the usability of the program by implementing accurate flow estimation and a more precise count. This would also open up the possibility of tracking individual groups of people if users wanted to keep an eye on a certain section of the crowd. However, going in this direction would likely also require higher information security, as you would now be tracking the movement of parts of the crowd.

References

Chan, Antoni, John Liang, and Nuno Vasconcelos. 2008. "Privacy Preserving Crowd Monitoring: Counting People Without People Models or Tracking." In *Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2008.4587569.

Contributors, PyTorch. 2023. *PyTorch Documentation*. https://pytorch.org/docs/stable/index.html.

Dahl, Sofie. 2023. "Crowd Management." EventSafety.

Feliciani, Claudio, Alessandro Corbetta, Milad Haghani, and Katsuhiro Nishinari. 2023. "Trends in Crowd Accidents Based on an Analysis of Press Reports." *Safety Science* 164: 106174. https://doi.org/https://doi.org/10.1016/j.ssci.2023.106174.

Fruin, John J. 1971. "Pedestrian Planning and Design."

Gao, Junyu, Maoguo Gong, and Xuelong Li. 2021. "Congested Crowd Instance Localization with Dilated Convolutional Swin Transformer." *CoRR* abs/2108.00584. https://arxiv.org/abs/2108.00584.

Gong, Jia-Yu. 2023. "Awesome-Crowd-Counting." https://github.com/gjy3035/Awesome-Crowd-Counting.

International Organization for Standardization. 2013. "ISO/IEC 27001:2013 - Information security management systems." 2013. https://www.iso.org/standard/27001.

Jenn, and Ken Visocky O'Grady. 2017. *A Designer's Research Manual: Second Edition, Updated + Expanded*. Second. Rockport Publishers.

Lempitsky, Victor, and Andrew Zisserman. 2010. "Learning to Count Objects in Images." In *Advances in Neural Information Processing Systems*, edited by J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta. Vol. 23. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2010/file/fe73f687e5bc5280214e0486b273a5f9-Paper.pdf.

Li, Bo, Hongbo Huang, Ang Zhang, Peiwen Liu, and Cheng Liu. 2021. "Approaches on Crowd Counting and Density Estimation: A Review." *Pattern Analysis and Applications* 24: 853–74.

Ma, Zhiheng, Xiaopeng Hong, and Qinnan Shangguan. 2023. "Can SAM Count Anything? An Empirical Study on SAM Counting." https://arxiv.org/abs/2304.10817.

Modolo, Davide, Bing Shuai, Rahul Rama Varior, and Joseph Tighe. 2021. "Understanding the Impact of Mistakes on Background Regions in Crowd Counting." In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 1650–59.

O'Shea, Keiron, and Ryan Nash. 2015. "An Introduction to Convolutional Neural Networks." *CoRR* abs/1511.08458. http://arxiv.org/abs/1511.08458.

"Python Documentation." 2023. https://docs.python.org/3/.

Raineri, Aldo. 2004. "The Causes and Prevention of Serious Crowd Injury and Fatalities at Outdoor Music Festivals." In. https://doi.org/10.13140/2.1.3036.0005.

SDU Cloud. 2020. "Introduction to Cloud Security." 2020. https://docs.cloud.sdu.dk/intro/security.html.

Simonyan, Karen, and Andrew Zisserman. 2014. "Very Deep Convolutional Networks for Large-Scale Image Recognition." *arXiv Preprint arXiv:1409.1556*.

Sommerville, I. 2011. *Software Engineering*. International Computer Science Series. Pearson. https://books.google.es/books?id=l0egcQAACAAJ.

Song, Qingyu, Changan Wang, Yabiao Wang, Ying Tai, Chengjie Wang, Jilin Li, Jian Wu, and Jiayi Ma. 2021. "To Choose or to Fuse? Scale Selection for Crowd Counting." *The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21)*.

Still, G. Keith. 2014. *Introduction to Crowd Science*. CRC Press.

The AI Learner. n.d. "OpenCV cv2.warpPerspective() Function." https://theailearner.com/tag/cv2-warpperspective/.

Zhang, Yingying, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. 2016. "Single-Image Crowd Counting via Multi-Column Convolutional Neural Network." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

12 Appendix

Technical specifications for camera

Teknisk specifikation til indsamling af data

Syddansk Universitet, Teknisk Fakultet, d. 10/7-2023

Teknisk specifikation til indsamling af data

Formålet med denne tekniske specifikation er at beskrive kravene til indsamling af videomateriale fra en festival til brug i et crowd safety management system. Projektet laves af studerende på Syddansk Universitet. Kontakt oplysninger:

Anton Irvold: anirv20@student.sdu.dk

Christoffer Krath: chkra19@student.sdu.dk

Krav til kameraer

- Kamera skal have tilstrækkelig opløsning til at kunne adskille mennesker fra hinanden (1080P og opad).
- Kamera skal være vidvinkel så fremt muligt.
- Kamera skal optage i 30 FPS så fremt muligt.

Krav til placering af kameraer

- Kameraer skal placeres på strategiske steder, såsom foran scener, indgange, udgange og områder med høj trafik.
- Kameraer skal placeres i en højde på mindst 3 meter eller højere for at undgå at blive blokeret af mennesker.
- Kameraer skal placeres i en vinkel, der muliggør dataindsamling af bredest muligt område.

Krav til dataindsamling

- Data skal gerne ligge på en harddisk eller cloud hvorfra det kan tilgås kort tid efter.

Figure 12: Technical specification document for cameras

Full class diagram for frontend

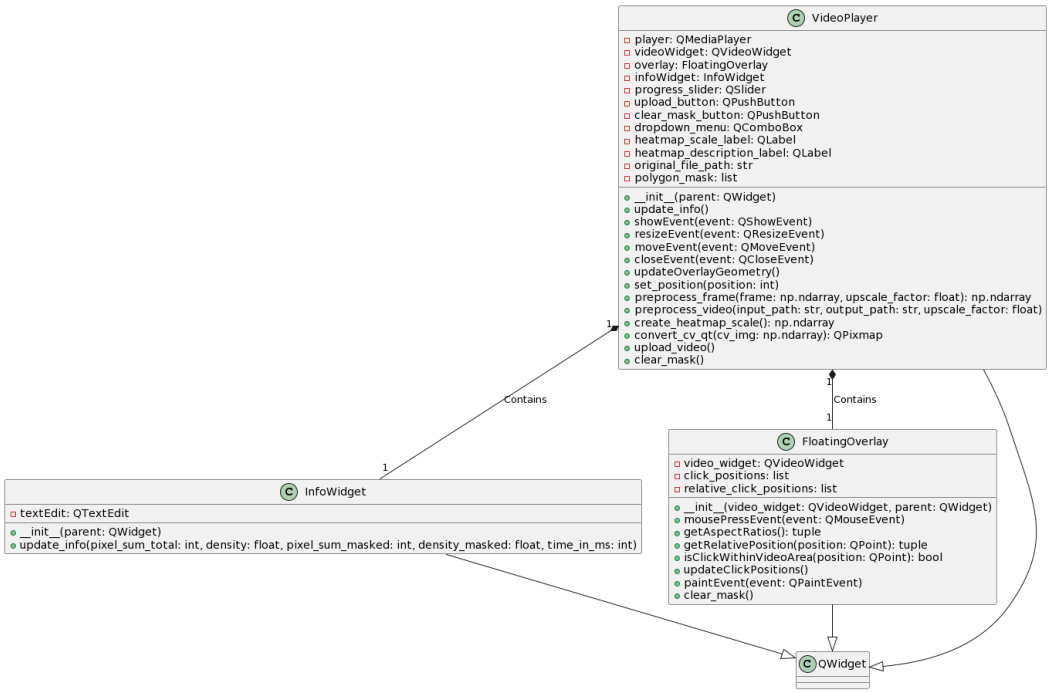


Figure 13: Class diagram for frontend

Timeline

Date	Activity
July-August	Gathering data at attended festivals in collaboration with Event Safety
August	Drafting the project description
31-08-2023	Delivering the project description
09-09-2023	Final discussion with Event Safety regarding the proposed solutions, requirements and the project going forward
September	In-depth analysis of use cases (with Event Safety) In-depth analysis of required technologies and methods used in academic literature. Design of the system
25-09-2023	Progress update with Event Safety regarding the design of the system
October	Implementation of proof of concept system (primarily backend)
26-10-2023	Progress update with Event Safety and small-scale user testing
31-10-2023	Attending Crowd Safety Course hosted by Event Safety in Copenhagen
November	Implementation of proof of concept system
December	Final implementation of the system Documentation of the system User testing and evaluation

Mentimeter presentation result

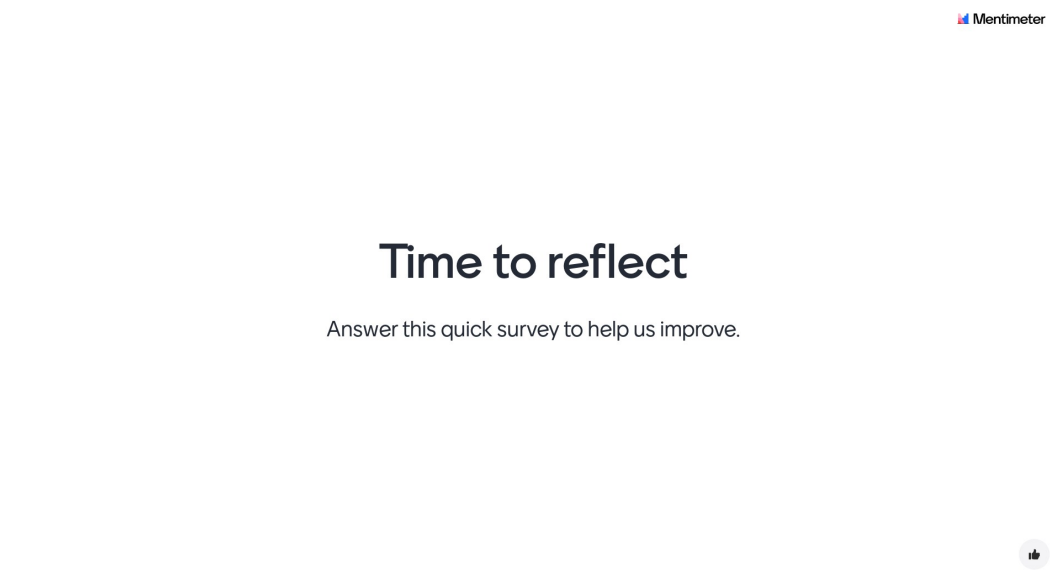


Figure 14: Mentimeter presentation result

Backend initialisation script

```
#!/bin/bash

git submodule init
git submodule update

sudo add-apt-repository -y ppa:deadsnakes/ppa
sudo apt-get update
sudo apt-get install -y libgl1-mesa-glx

# Install Python 3.7 and the distutils package which is required for pip installation
sudo apt-get install -y python3.7 python3.7-distutils

# Set Python3.7 as the default python3 installation
sudo update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.7 1
sudo update-alternatives --config python3

# Install pip for Python3
sudo apt-get install -y python3-pip

# Upgrade pip to the latest version
python3 -m pip install --upgrade pip

# Install packages from the requirements file
pip install -r CrowdCounting-SASNet/requirements.txt
pip install -r requirements.txt
```

Crowd Counting Documentation (technical users)

12.0.1 System requirements

- Debian-based Unix system like Ubuntu.
- Able to execute Bash scripts with root access.
- Git installed.
- Internet access.

12.0.2 Step-by-step guide

1. Clone the repository at https://github.com/anirv20/crowd-safety.
2. Run the init.sh using `bash backend/init.sh`.
3. Download the SHHA.pth model weights from SASNet.
4. In main.py: Configure the path to the SHHA.pth model weights. (MODEL_PATH)
5. In main.py: Configure the output directory. (OUTPUT_DIR)
6. In main.py: Configure the frame sampling interval. (FRAME_INTERVAL)

7. In main.py: If the intended use is with the frontend: Set UPSAMPLING_FACTOR to 1 and COLOR_MAP to None.
8. In main.py: Define the corners of the cropped area on the video in (x, y) pairs in local_coordinates for each Camera instance.
9. In main.py: Define the global coordinates based on the land survey for each Camera instance.
10. In main.py: Define the path to the camera video for each Camera instance.
11. In main.py: Add the Camera instance to the same or different Camera collections.
12. Run main.py and download the output video from the specified OUTPUT_DIR. This can now be uploaded to the frontend if step 7 was followed.

Frontend Guide (Non-technical users)

Below is a user guide for the frontend and its functions

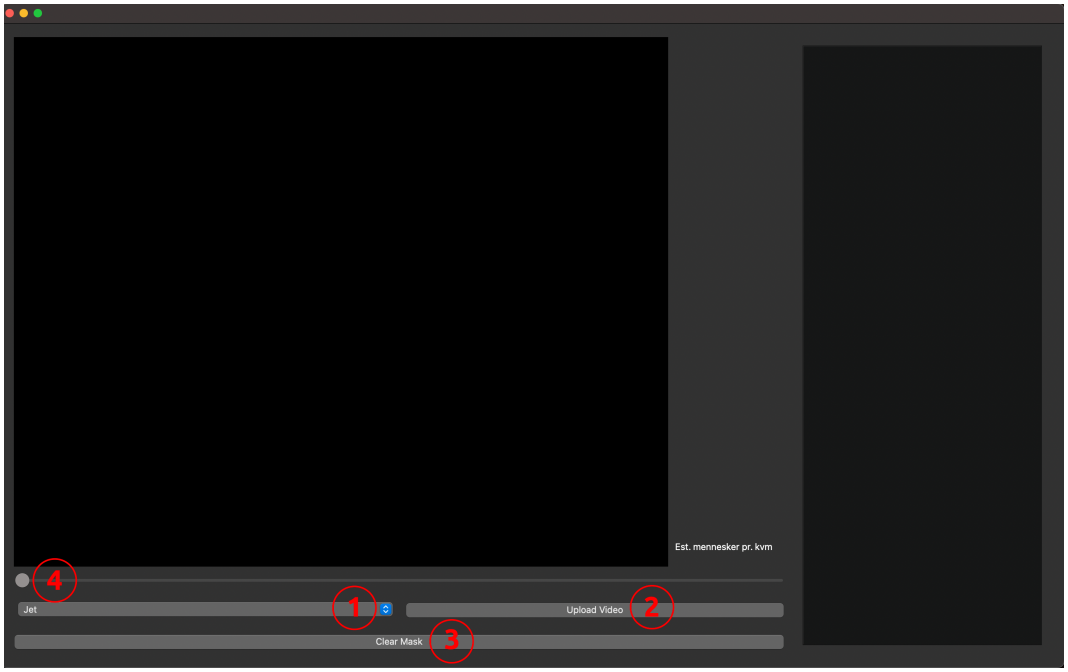


Figure 15: Frontend user guide

1. Colorspace for the heatmap. This menu allows the user to select a specific color space for the heatmap. The colorspace "JET" is selected by default
2. The upload video button allows the user to upload a video that has been processed by the backend
3. When a video has been uploaded, the user can use their mouse to select a series of points to define an area for specific analysis - See Figure 9. This button removes the polygon and allows the user to mark a new one.

4. When a video is uploaded, the user can use the progress bar to move forward or backward in the video.