

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

Sune Lundø Sørensen

University of Southern Denmark,
MMMI
suso@mmtmi.sdu.dk

Supervisor

Christian Sejlund

Event Safety (Smukfest & Muskelsvindfonden)
christian@eventsafety.dk

Company Collaborator

Christoffer Krath

University of Southern Denmark,
Technical Faculty
chakra19@student.sdu.dk

Author

Mikkel Baun Kjærgaard

University of Southern Denmark,
MMMI
mbkj@mmtmi.sdu.dk

Supervisor

Sofie Dahl

Event Safety (Smukfest & Muskelsvindfonden)
sofie@eventsafety.dk

Company Collaborator

ABSTRACT

In spring 2023 we reached out to Event Safety A/S regarding the possibility of a collaboration with them regarding our final project of the bachelor of engineering in software technology.

The idea was a software system using 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 we needed supervision for our project 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 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 completed, manual count comparisons, a thourough user test through a focus group was held for other 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. The system has precision that makes it useable to the inteded user while allowing for further development of more functionality. The system can successfully be leveraged by safety experts to improve crowd safety and crowd safety management

Table of contents

1 Acronyms and definitions	5
2 Introduction	6
2.1 Preface	6
2.2 Context	6
2.3 Problem Statement	8
2.4 Planning	8
3 State of the art	8
3.1 Convolutional Neural Network (CNN) for Computer Vision	8
3.2 Solving Scale-Variation with SASNet	8
4 Analysis	9
4.1 System specification	9
4.2 Functional Requirements	10
4.3 Non-functional requirements	12
4.4 Risk analysis	13
4.5 Risk assessment	14
5 Design	16
5.1 About the choice of crowd counting	16
5.2 Design considerations regarding security & privacy	17
5.2.1 Facial recognition vs. Crowd Counting	17
5.2.2 UCloud data policy	17
5.3 Choosing a technology for Crowd Counting	18
5.4 Backend behavior	18
5.5 Design considerations regarding user experience	20
5.6 The Cyber-Physical System	20
6 Implementation	22
6.1 Data Collection and Preparation	22
6.2 Backend implementation	22
6.2.1 Environment setup	24
6.2.2 UCloud service	24
6.2.3 SASNet Model Adaptation	24
6.2.4 Implementation of requirements	24
6.3 Frontend	29
6.4 Integeration	29
6.5 Optimization and Performance	29
7 Validation and Verification	31
7.1 Accuracy of Crowd Count	31
7.2 Runtime and GPU usage	31
7.3 Event Safety presentation	32
7.3.1 Focus group	32
7.3.2 Focus Group Questionnaire	33
7.4 Further evaluation of functional requirements	35
7.5 Further evaluation of non-functional requirements	36

8 Discussion	38
9 Conclusion	40
10 Future Perspectives	41
11 Appendices	42
11.1 Technical specifications for camera	42
11.2 Full class diagram for frontend	44
11.3 Timeline	45
11.4 Mentimeter presentation result	46
11.5 Backend initialisation script	47
11.6 CROWD COUNTING DOCUMENTATION (technical users)	47
11.6.1 System requirements	47
11.6.2 Step-by-step guide	47

1 Acronyms and definitions

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 et al., 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 own actions.
Mosh pit	A crowd situation where people vacate an area in the crowd followed by violent or aggressive dancing. Sizes of Mosh pits vary greatly.
Technical user	A developer or technician that 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

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 to the problem and an introduction to the project collaborators.

2.1 Preface

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.

Event Safety, Sofie Dahl and Christian Sejlund for their time, expertise and knowledge during the analysis, design and implementation phases. Also a big thanks to the rest of Event Safety, Smukfest and Muskelsvindfonden employees for all feedback.

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 aerial videos during Grøn Koncert.

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

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

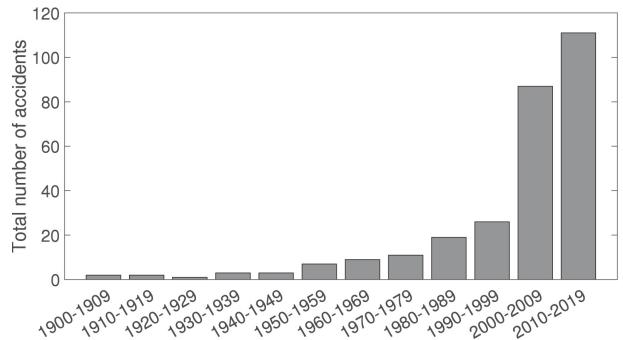
2.2 Context

As seen from Figure 1 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.

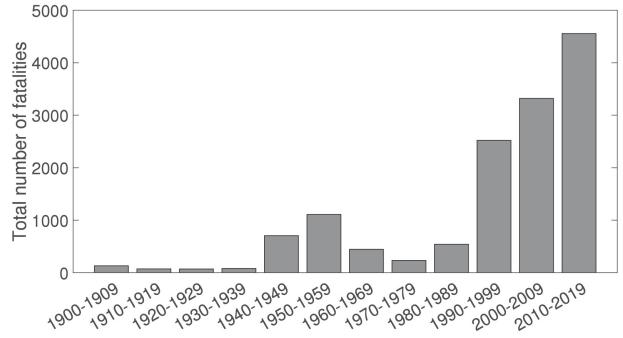
It is worth mentioning that these figures and descriptions of situations are often from media outlets. As such, it is not the full story. While these statistics might make it seem that general safety and incidents at festivals and the like have gotten worse, it is the opposite. It is generally incredibly 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.

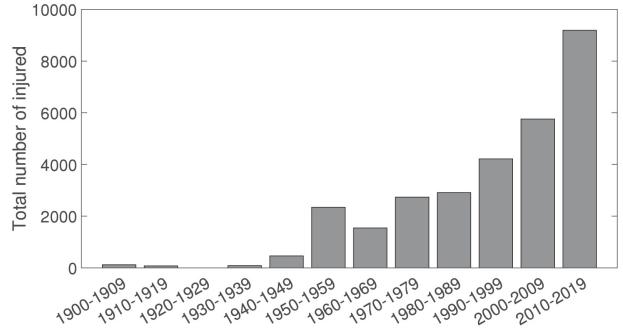
To have the best possible outcome for this project, Event Safety invited us to Grøn Koncert and Smukfest in July and August to see how they work and gather security video footage of real crowds at festivals to use as training data. We will also use Event Safety for their practical and theoretical knowledge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



(a) Accidents by decade



(b) Fatalities by decade



(c) People injured by decade

Figure 1: Fatalities in crowds graph (Figure from Feliciani et al., 2023)

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

The project is going to follow a waterfall method because of the group size and timeframe. The exact timeline can be seen in Section 11. The official project start is in september. Before this we will have prepared the project description, company collaboration, and collected data. September will focus on analysis and design of the system, October and November will contain the implementation phase and the testing will be done in the beginning of December. The project report will be written continuously.

3 State of the art

This section will describe the state of the art technologies from academic litterature 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 Networks (ANN) that are used to “solve difficult image-driven pattern recognition tasks.” (O’Shea & Nash, 2015) ANN’s use input in the form of a multi-dimensional vector, which will be distributed to a number of 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.

CNN’s are similar to ANN’s in almost every way, with the only difference being that CNN’s allow to encode 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 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 2.

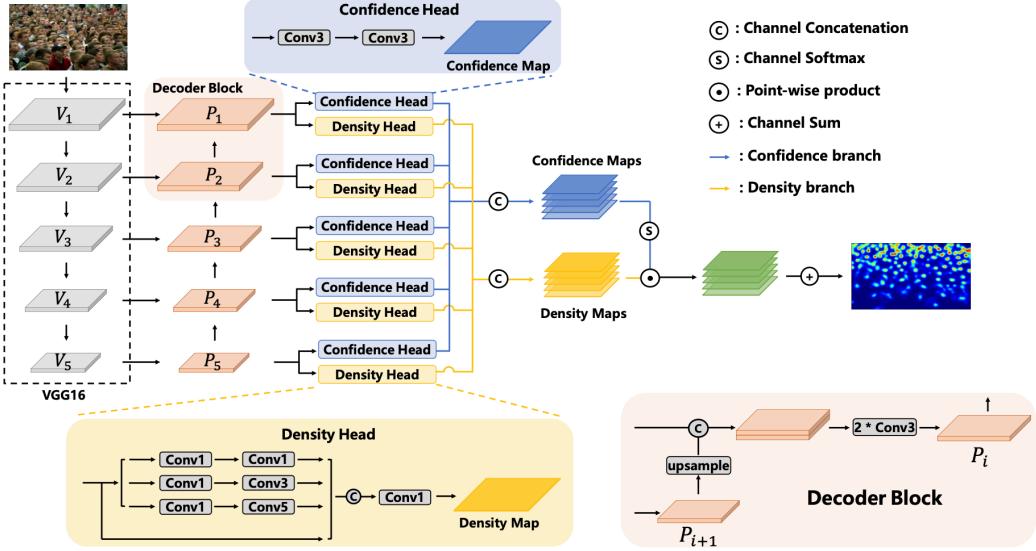


Figure 2: Architecture of the SASNet model (Diagram from Song et al., 2021)

As it can be seen from the diagram, SASNet uses the first 13 convolutional layers from VGG-16 (Simonyan & 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.

4 Analysis

This section 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 recognise 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.

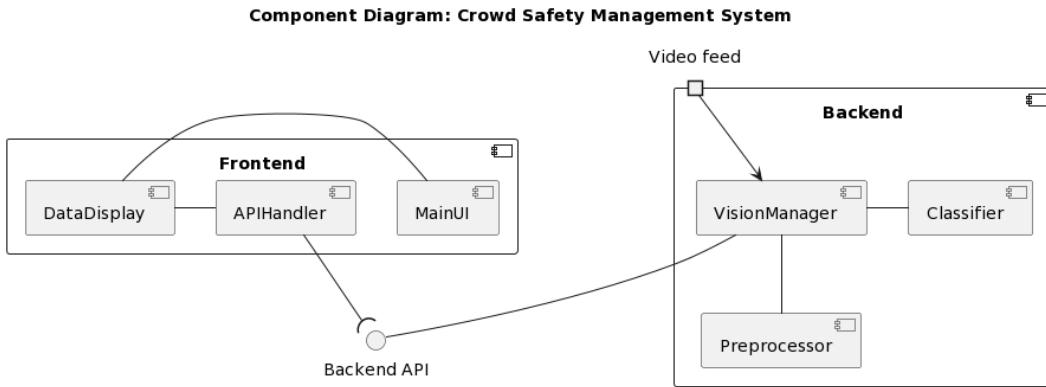


Figure 3: Component diagram of the system

An initial component diagram that maps this described system but is subject to change can be seen in Figure 3.

4.2 Functional Requirements

This section describes the functional requirements for the proposed system along with their “MoSCoW” prioritisation (“must have”, “should have”, “could have”, “will not have”). These are shown in Table 2.

ID	MoSCoW	Requirement
1	M	The system must be able to count the amount 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

ID	MoSCoW	Requirement
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 other) 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 surge, 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 2: Table of functional requirements

4.3 Non-functional requirements

This section describes the non functional requirements for the proposed system along with their “MoSCoW” prioritisation (“must have”, “should have”, “could have”, “will not have”). These are shown in table Table 3.

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 3: 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 are able to mitigate some of these risks. Table 4 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 supervisor if alternatives exist.
R03	Personal Conflicts	Product, project	A project related or personal conflict between the 2 members of the project having an effect on either 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 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 4: Table of risks and their impact

4.5 Risk assessment

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

$$Effect = Probability * Severity$$

The probability and severity is 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 5.

ID	Probability	Severity	Effect	Notes
R01	1	3	3	The severity is based on the amount of code lost.

ID	Probability	Severity	Effect	Notes
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 are able to adapt to it.
R05	1	5	5	Severity would depend on when in the process Event Safety would cancel our collaboration.

Table 5: Table of risk assesment

5 Design

This section is going to describe the design choices made for the system described in the analysis of the problem and specified in the requirement specification.

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 et al. (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 et al., 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 from using crowd counting rather than object detection is:

- 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 of specific training data.
- Crowd counting is not as exposed to the risk of over fitting the model. (Li et al., 2021)
- Crowd counting does not single out individuals, but looks at the crowd holistically. (Chan et al., 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 density of concert crowds.

The paper Li et al. (2021) says the current leading approach to crowd counting is the use of a CNN. This paper, contrary to our definition of crowd counting, includes object detection as a method of crowd counting. It also includes regression based methods and, of course, CNN's. The main challenge with CNN's for crowd counting is to differentiate between the large scale variations in the countable human heads. Li et al. (2021) discuss several categories of approaches for CNN based models to solve the 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 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 purposes of the CSMS test data, “ShanghaiTech Part A” is a more representative dataset to use, since it contains more densely populated crowds from a birds 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 & 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 with 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 movement of members of the crowd, rather than using a detection based counting method. Despite of 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 this method.

5.2 Design considerations regarding security & 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 anonymises the output data and does not track individuals through time, which could be a privacy concern.

5.2.2 UCloud data policy

As mentioned, data handling and data privacy are very important for this project. Due to this, we had 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 were able to 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 pretrained model weights (trained on ShanghaiTech A and B), which means we can save many compute- 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 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 4 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 visualise a rough partitioning into classes and for better readability throughout the diagram. The input is specified as the path to the pretrained model and the path to the video for it to run on.

Below is a short runthrough of the different partitions. A more in-depth description can be found in the implementation section

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 VGG16 model which it is based on. This sets the foundation for subsequent image processing.

3. Pre-processing of input data.

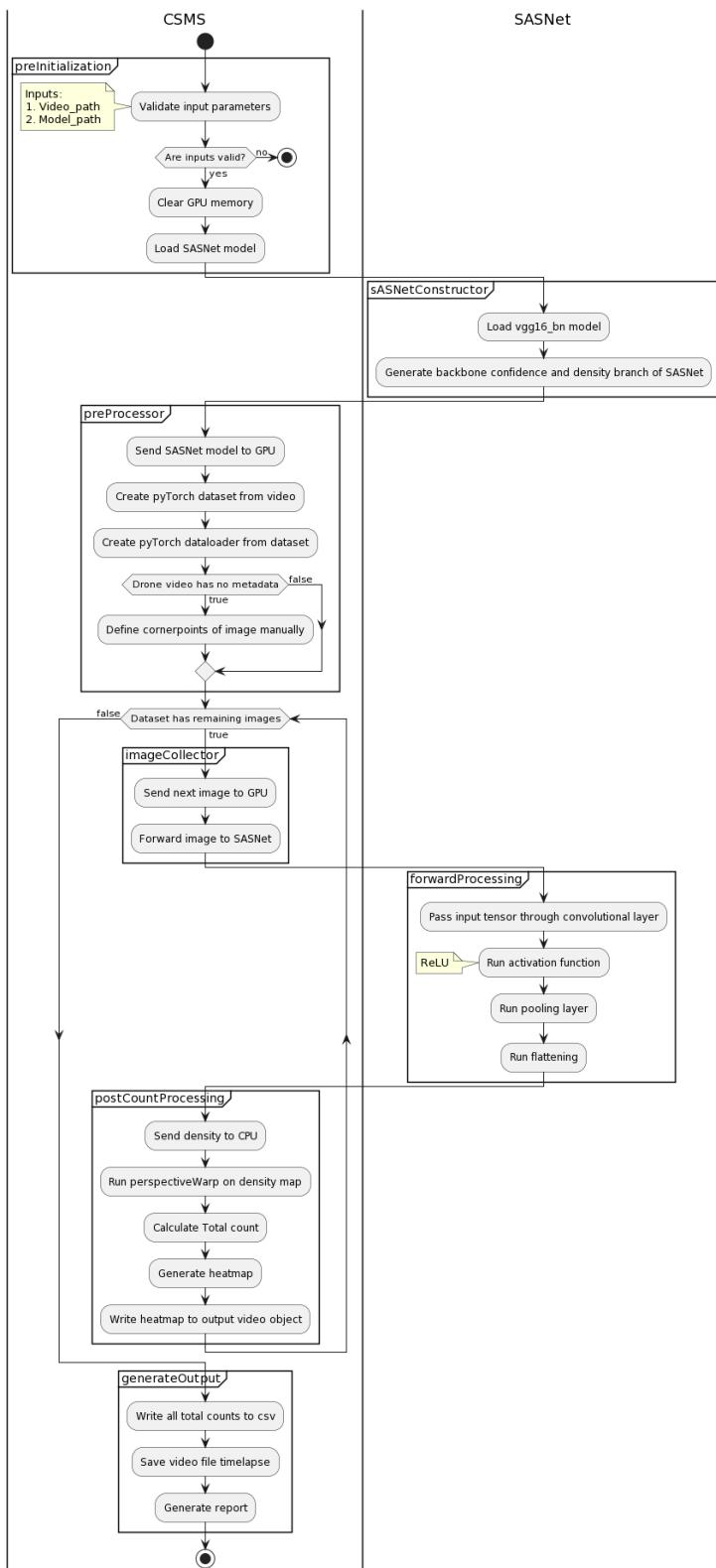


Figure 4: Activity diagram depicting flow in the solution

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, 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 before the program terminates.

5.5 Design considerations regarding user experience

For many purposes a video with a heatmap could be sufficient. However, in order to make it possible to satisfy requirements such as functional requirement 2 and 8 in Table 2 as well as non-functional requirement 2 in Table 3, 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, in order to analyse the data further, described as a “reporting tool” in functional requirement 8. The frontend should also 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 1 pixel to 1 square meter. This can be upscaled in the frontend again to increase the image 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 on service levels. 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 worrysome. (Dahl, 2023)

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 10.

It is necessary to implement an IoT framework to implement non-functional requirement 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 prioritisation.

6 Implementation

This section is going to describe the implementation of as many designed elements as possible within the timeframe, following the prioritisation 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 the purposes of 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 10 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 11.

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 5.

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

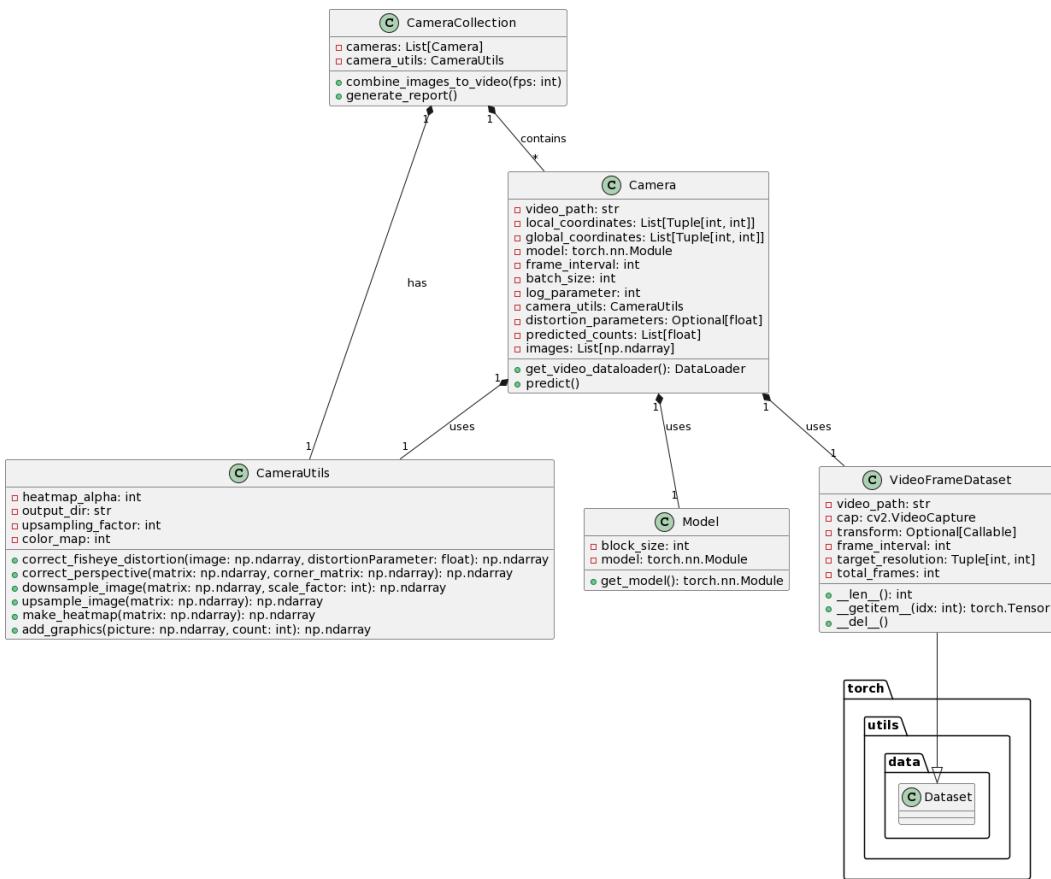


Figure 5: Class diagram depicting the structure of the backend

6.2.1 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 $\geq 1.5.0$ (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 11 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

6.2.2 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 6.2.1 is run to initialise the environment.

6.2.3 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.

6.2.4 Implementation of requirements

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

6.2.4.1 Implementing crowd counting This section describes the implementation of functional requirement 1: “*The system must be able to count the amount of people in the crowd with a 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 de

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

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

            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(downscaled_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 is 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 6.2.4.2. 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.

Måske billede af processen som billedet gennemgår her?

6.2.4.2 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 6.

The scale factor will be mentioned later on in this section.

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 7.

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.

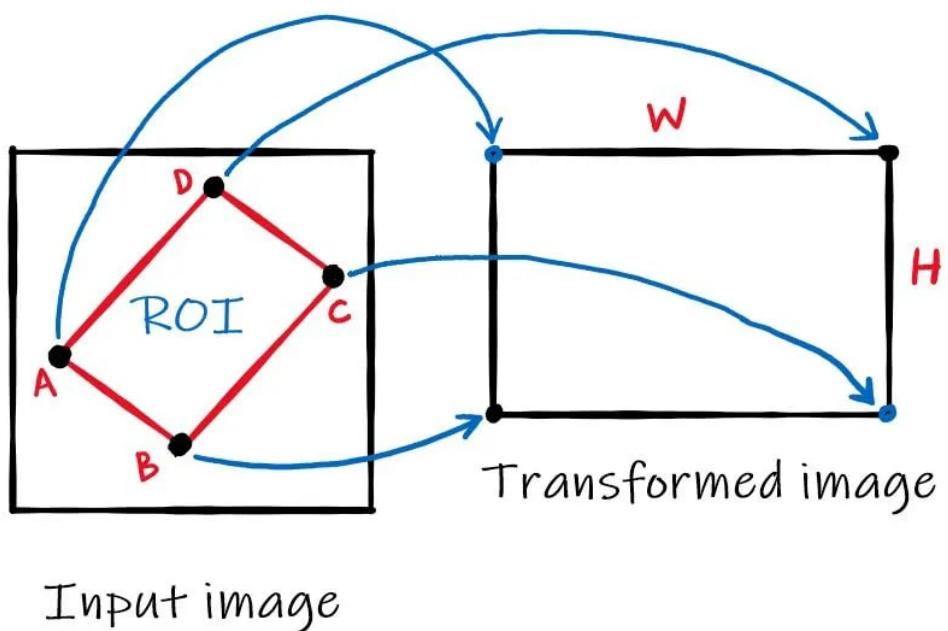
This leaves us with the steps condensed into:

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. (Is only done if the frontend is not used)

$$\begin{bmatrix} t_i x' \\ t_i y' \\ t_i \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & b_1 \\ a_3 & a_4 & b_2 \\ c_1 & c_2 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

↳ **Scaling Factor** ↳ **Transformation Matrix (M)**

Figure 6: Depiction of transformation matrix and scale factor (From The AI Learner, n.d.)



Input image

Figure 7: Depiction of perspective correction from The AI Learner (n.d.)

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.

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 have access to the Python OpenCV module to allow for preprocessing of the video. There are 3 main widgets in the frontend. The main VideoPlayer widget which has the buttons to upload the video, clear mask and change heatmap colorscheme. The InfoWidget which displays textual data such as the count, count in selection, density, density in selection and time passed. The FloatingOverlay which 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 2.

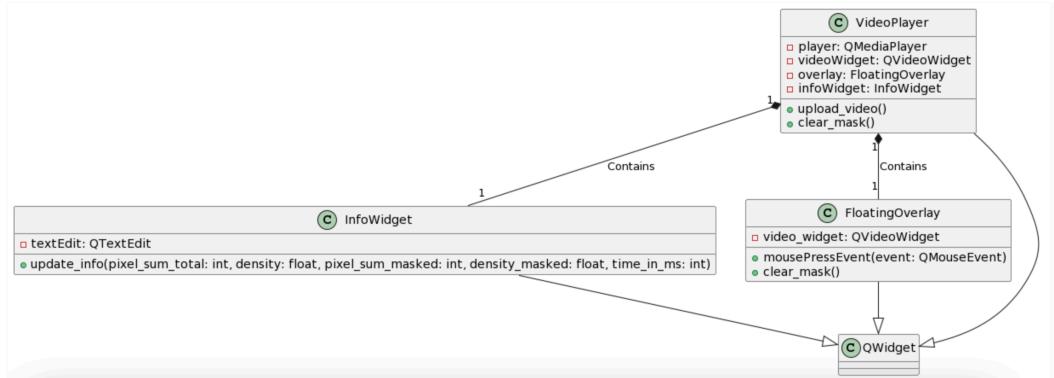


Figure 8: Simplified class diagram depicting the structure of the frontend

A simplified version of this structure can be seen in Figure 8 and the full diagram with all methods and attributes can be seen in the appendix. The VideoPlayer calls a preprocess function which 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 2.

6.4 Integeration

Although a proper API between the frontend and backend was described in the component diagram, Figure 3, it has not been a priority of the project and thus not 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 in order to be processed 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 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 Optimization 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. 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 bottle neck.
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 reduced the 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 prediction_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 the 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 9: Small section of bar after concert - Manual count

In the image seen in Figure 9, 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 10.

By looking at the data in the infobox, we see that the model predicted the area to contain 119 people. Using this result, we can conclude that in areas where a human is able to somewhat accurately count a crowd, the model is able to do the same fulfilling functional requirement 1: “The system **must** be able to count the amount of people in the crowd with a precision required by the user base.”

7.2 Runtime and GPU usage

While GPU usage is something already discussed in implementation, runtime and the optimization of this is 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

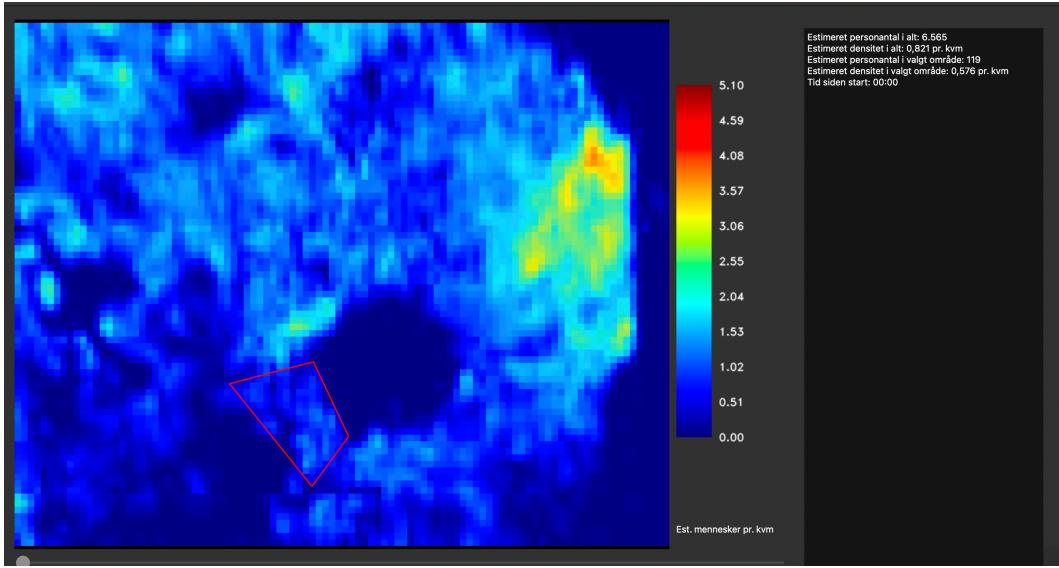


Figure 10: Small section of bar after concert - Model count

this live, additional resources or optimisation 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 10.

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 & Visocky O'Grady, 2017). This fits the need 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 others interests if one believe there are 2.000 people in an area, while another believe there are 12.000 people.

7.3.2 Focus Group Questionnaire

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 11.

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 prioritisation 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

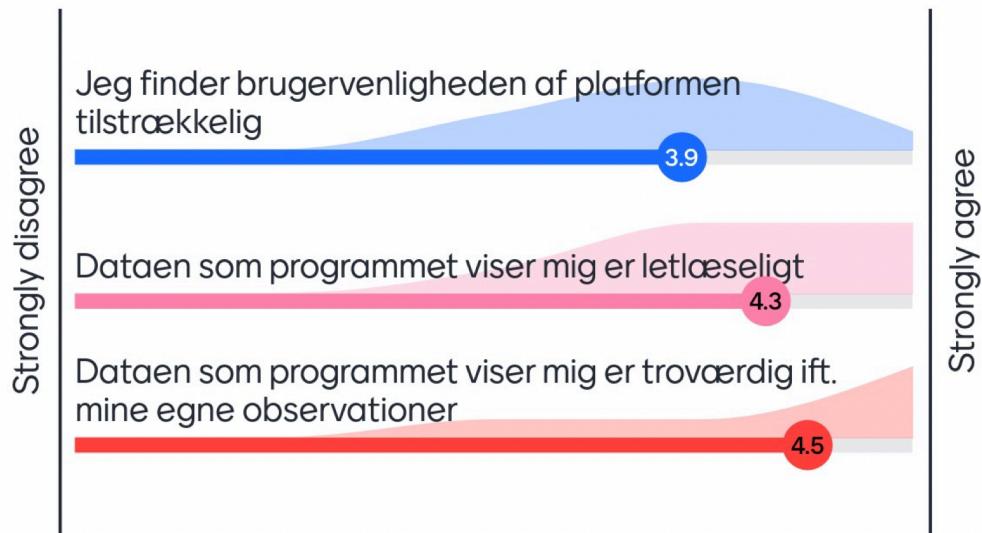


Figure 11: Questionnaire question 2

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 12: 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 varies depending on the total size of the crowd.

7.4 Further 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 a 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 summarizing 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 Further 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 11.

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 11.

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 property 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 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 systems 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 on the choice of technologies.

In order to evaluate on 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 is trained on, SASNet performs with a mean average error of 53.59 (Song et al., 2021). According to Modolo et al. (2021), the Shanghai Tech 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 own 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 is 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 on 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 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. Number 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 requirement 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 on the non function requirements in Section 7.5, the “must have” and “should have” requirement 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 of 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 was also one of the reasons why we decided to use the CNN 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 extend 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 a counting by detection algorithm, making it unlikely that it has a large impact on the final result. However the choice of crowd counting method made the implementation of some functional requirements, in regards to movement of crowd members, more difficult to implement which resulted in them not being implemented. A work around could be implemented while still using SASNet by looking at delta changes in densities over time 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, that was not the case. The prioritization 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 is 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 depends on the knowledge of the earlier stages. Evaluating on this choice, it has been a good process for the project since we ended up with a finished and working product. A Scrum process could have resulted in difficulty to meet deadlines because of the group size of 2.

9 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 reliability at the festivals Grøn and Smukfest. We tested the system at these 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 using aggregate sums of people per square meter, showing our commitment to ethical data handling.

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 CNN-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. Environmental factors like poor lighting and stage effects sometimes affected how accurately the system worked. These are common issues at festivals and made 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 not features that were missing for achieving the overall goal. 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.

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 crowd safety 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 summarized in section Section 10.

10 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 8 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 for 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 organization in charge of large gatherings people. This could be other users like police, defence authorities, emergency management agencies, transportation authorities, sport stadium managers, retail and shopping malls, theme park managers, for handling events like protests, public national celebrations (eg. sport celebrations, royal birthdays), sport 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 view point, and a group of people with the necessary knowledge on 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 venus 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 setup to allow for this, by defining multiple cameras and their physical bounds.

It could be an interesting perspective to cooperate more cyber-physical or IoT aspects 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 the 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 predict 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.

11 Appendices

11.1 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 oplosning 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 13: Technical specification document for cameras

11.2 Full class diagram for frontend

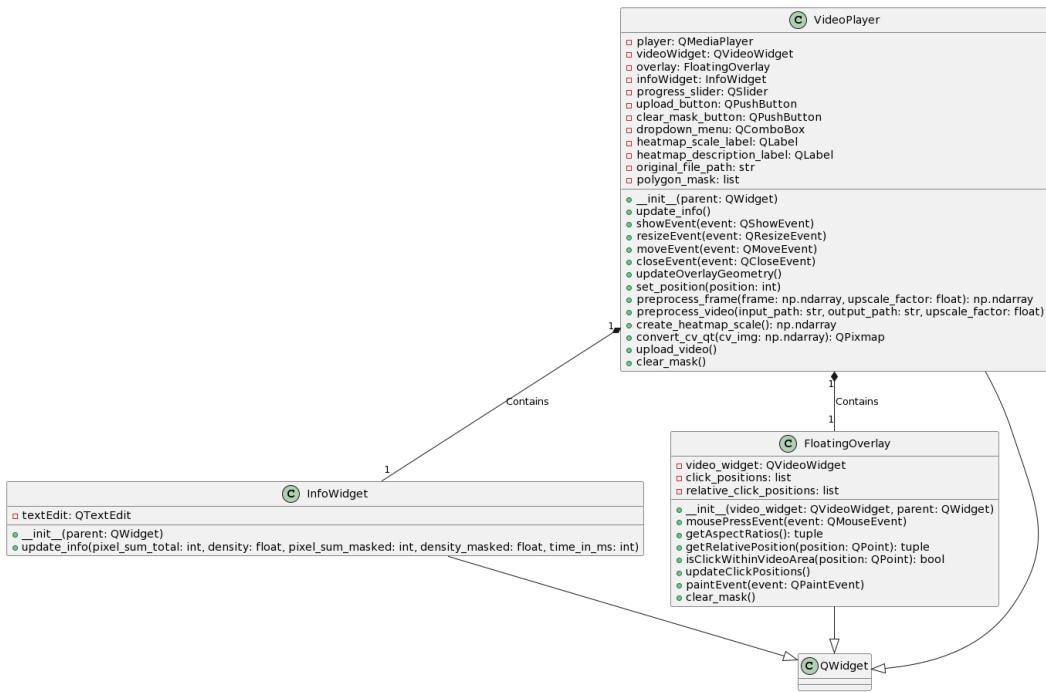


Figure 14: Class diagram for frontend

11.3 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

11.4 Mentimeter presentation result

 Mentimeter

Time to reflect

Answer this quick survey to help us improve.



Figure 15: Mentimeter presentation result

11.5 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
```

11.6 CROWD COUNTING DOCUMENTATION (technical users)

11.6.1 System requirements

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

11.6.2 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.

References

- Chan, A., Liang, J., & Vasconcelos, N. (2008). Privacy preserving crowd monitoring: Counting people without people models or tracking. *Computer Vision and Pattern Recognition*. <https://doi.org/10.1109/CVPR.2008.4587569>
- Contributors, P. (2023). *Pytorch documentation* [Version 2.1.0]. <https://pytorch.org/docs/stable/index.html>
- Dahl, S. (2023, October). Crowd management [Course instructor: Sofie Dahl, Security Advisor. Duration: 09:00 to 16:00].
- Feliciani, C., Corbetta, A., Haghani, M., & Nishinari, K. (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>
- Gong, J.-Y. (2023). Awesome-crowd-counting [GitHub repository].
- International Organization for Standardization. (2013). *ISO/IEC 27001:2013 - Information security management systems* [Accessed: 1-12-2023]. <https://www.iso.org/standard/27001>
- Jenn & Visocky O'Grady, K. (2017). *A designer's research manual: Second edition, updated + expanded* (Second) [Paperback edition published in 2009]. Rockport Publishers.
- Lempitsky, V., & Zisserman, A. (2010). Learning to count objects in images. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, & A. Culotta (Eds.), *Advances in neural information processing systems* (Vol. 23). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2010/file/fe73f687e5bc5280214e0486b273a5f9-Paper.pdf
- Li, B., Huang, H., Zhang, A., Liu, P., & Liu, C. (2021). Approaches on crowd counting and density estimation: A review. *Pattern Analysis and Applications*, 24, 853–874.
- Ma, Z., Hong, X., & Shangguan, Q. (2023). Can sam count anything? an empirical study on sam counting.
- Modolo, D., Shuai, B., Varior, R. R., & Tighe, J. (2021). Understanding the impact of mistakes on background regions in crowd counting. *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 1650–1659.
- O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. *CoRR*, *abs/1511.08458*. <http://arxiv.org/abs/1511.08458>
- Raineri, A. (2004). The causes and prevention of serious crowd injury and fatalities at outdoor music festivals. <https://doi.org/10.13140/2.1.3036.0005>
- SDU Cloud. (2020). *Introduction to Cloud Security* [Accessed: 1-12-2023]. https://docs.cloud.sdu.dk/intro_security.html
- Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- Song, Q., Wang, C., Wang, Y., Tai, Y., Wang, C., Li, J., Wu, J., & Ma, J. (2021). To choose or to fuse? scale selection for crowd counting. *The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21)*.
- The AI Learner. (n.d.). *OpenCV cv2.warpPerspective() Function* [Accessed: October 2023]. <https://theailearner.com/tag/cv2-warpperspective/>
- Zhang, Y., Zhou, D., Chen, S., Gao, S., & Ma, Y. (2016). Single-image crowd counting via multi-column convolutional neural network. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.