Access the code, data, and analysis at github repo

Using computer vision and AI techniques to improve crowd safety management

Project report - January 2024

Anton Irvold
University of Southern Denmark
anirv20@student.sdu.dk

Christoffer Krath
University of Southern Denmark
chkra19@student.sdu.dk

Administrative Info

Collaborators from Event Safety

- Sofie Dahl, sofie@eventsafety.dk
- Christian Sejlund, christian@eventsafety.dk

SDU supervisor

- Sune Lundø Sørensen, slso@mmmi.sdu.dk

Context and Background

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. This 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. Sudden panic in a crowded place can thus affect the safety of the crowd.

It is worth mentioning that these figures and descriptions of situations are from media outlets. As such, it's 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 newer technologies we will attempt to enhance it 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.

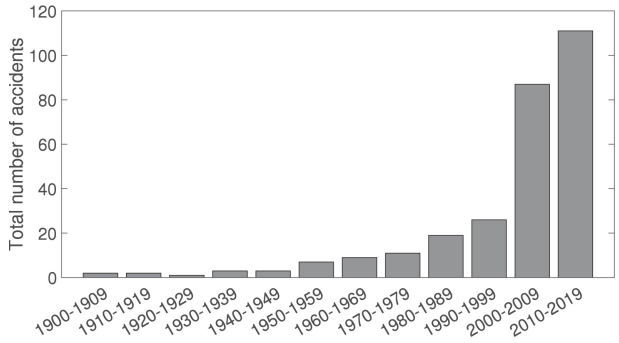
Problem Statement

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?

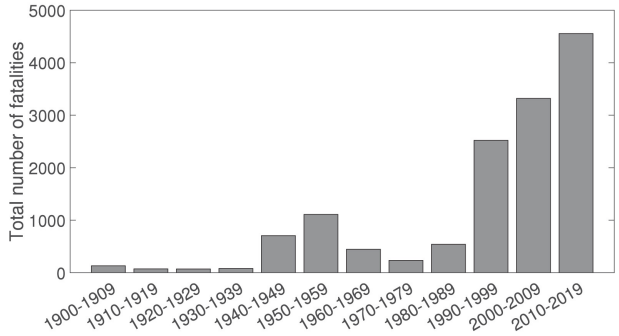
Timeline

Date	Activity
July-August	Gathering data at attended festivals in collaboration with Event Safety

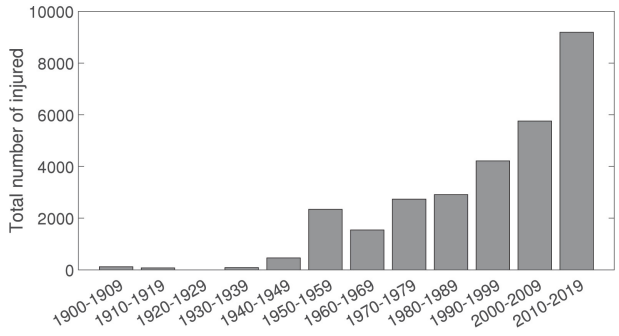
Date	Activity
August	Drafting the project description
31-08-2023	Delivering the project description
09-09-2023	Final discussion with Event Safety regarding the proposed solutions, requirements and the project going forward
September	In-depth analysis of use cases (with Event Safety) In-depth analysis 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



(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)

Analysis

Acronyms and definitions

Acronyms	Definition
CSMS	Crowd Safety Management System
Crowd Counting	The defintion of crowd counting in this report follows the definition from Chan et al. (2008): "[…] a privacy-preserving system for estimating the size of inhomogeneous crowds, […] without using explicit object segmentation or tracking"
AI	Artificial Intelligence
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
ML	Machine Learning
CCTV	Closed Circuit Television
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

Acronyms	Definition
SASNet	Scale-adaptive selection network Song et al. (2021)

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 live footage from these cameras, uses the model to segment the crowd, and learns what a dangerous situation might look like. 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 developed in either Java or Python, with an implementation of a machine learning model to handle the video feed according to the bullet list above and exposing this data through an API
2. A simple frontend or GUI (Graphical User Interface) 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.

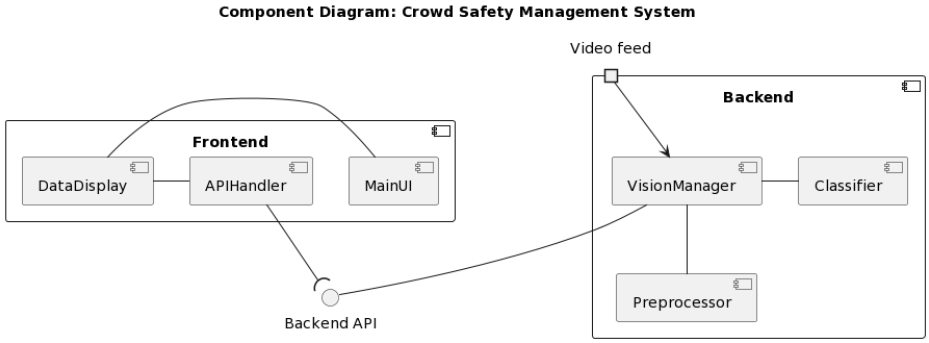


Figure 2: Component diagram of the system

An initial component diagram that maps this described system can be seen in Figure 2.

Functional Requirements

ID	MoSCoW	Requirement
1	M	The system **must** be able to count the amount of people in the crowd with a precision of <100 MAE
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 "rea" density pr. area unit
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 summarizing report of the concert with statistics of (crowd density, crowd count, risk factors, choke points, and other) relevant safety indicators 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

Non-functional requirements

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

Detailed Requirements

Risks

Method and theory

Convolutional Neural Network (CNN)

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 function on an input in the form of a multi-dimensional vectors, 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.

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, density or type of object. (Song et al., 2021) This differentiates it from object detection which is 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.

(Li et al., 2021) says the current leading approach to crowd counting is the use of a CNN. This paper, contrary to our definiton 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, "Shagnhaitech" comes from the paper Zhang et al. (2016). In this paper they have 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 analoguous 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 purpose of the CSMS.

SASNet

Scale variation is one of the main challenges with crowd counting, since perspective in the image will lead to different size of heads. One solution to this, SASNet, is proposed in Song et al. (2021). SASNet relies on the fact that heads in a local path share roughly the same scale.

Multi-scale fusion for SASNet.

Regarding SAM model for crowd counting:

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 annota- tions, which could hinder its ability to differentiate between different objects. Nevertheless, further exploration of adapt- ing SAM to the object counting task is still worth studying." (Ma et al., 2023)

Completely Self-Supervised Crowd Counting via Distribution Matching
Unsupervised Learning, Crowd Counting, Deep Learning

Possible to use a model not trained on relevant data but still able to produce meaningful results. Some are even better than other models. (Sam et al., 2020)

Design

Implementation

Initial proof of concept

Setup an environment implemeneting SASNet. Eliminates use of anaconda but perhaps at the cost of a paid subscription.

It was consideret to use a CNN model called CRSNet. However, reasarch suggest that CRSnet might have problems with scale variations, which is a common problem in crowd counting. Because of this, SASNet is quite possible a much better solution. (**sasnetCrowd**)

The first implementation was setup in Google (**Colab**). Colab provides free access to CUDA powered GPU power, which is needed for the implementation as it relies on tensorflow and pyTorch. specifically, pyTorch needs CUDA to function. Our implementation is based on an opensource implementation of (Song et al., 2021) found here.

The introductory parts of the implementation revolving cloning the directory and setting up google drive with the data will not be discussed, only the python code itself:

After installing the correct requirements, the pretrained model from SASnet is loaded into colab and the image to run the model on is loaded into the program:

```
def predict(args):
  img_path = args.image_path
  transform = standard_transforms.Compose([
      standard_transforms.ToTensor(), standard_transforms.Normalize(mean=[0.485, 0.456, 0
                                                                     std=[0.229, 0.224, 0.
  ])
    # open the image
  img = Image.open(img_path)
  return perspective_correction(img).convert('RGB')
  img = transform(img)
  img = torch.Tensor(img)
  img = img.unsqueeze(0)
  img = img.cuda()

  print(img.shape)

  model = SASNet(args=args).cuda()
  # load the trained model
  model.load_state_dict(torch.load(args.model_path))
  print('successfully load model from', args.model_path)
```

Following this, the predicted density map and color channels are extracted. The “jet” colorspace is then added to the density map to create a heatmap which in turn is layered onto the original image with a lowered opacity.

```
with torch.no_grad():
  model.eval()

  # get the predicted density map
  pred_map = 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]) / args.log_para
      print(f'Predicted Count: {pred_cnt}')

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

      # Normalize the grayscale image to the 0-255 range
      normalized_map = cv2.normalize(grayscale_map, None, 0, 255, cv2.NORM_MINMAX)

      # Apply the 'jet' colormap
      colormap = cv2.applyColorMap(np.uint8(normalized_map), cv2.COLORMAP_JET)

      # Add text to the heatmap
      cv2.putText(colormap, f'Predicted Count: {pred_cnt}', (50, 50),
                  cv2.FONT_HERSHEY_SIMPLEX, 1,
                  (0, 255, 0), 2)

      # Read the original image
      original_image = cv2.imread(img_path)  # Replace 'your_original_image.jpg' with y

      # Resize the heatmap to match the original image's size
      colormap_resized = cv2.resize(colormap, (original_image.shape[1], original_image.

      # Set the opacity (alpha) value
      alpha = 0.75  # You can adjust the opacity as needed

      # Blend the heatmap and original image
      blended_image = cv2.addWeighted(original_image, 1 - alpha, colormap_resized, alpha

      os.chdir("/content/gdrive/MyDrive/crowd-counting-data/output")

      cv2.imwrite('overlayed_image3.jpg', blended_image)

print("Predict Over")
```

Finally, the model is run on our image:

```
class Args:
  model_path = "../../gdrive/MyDrive/crowd-counting-data/SASNet/models/SHHB.pth"
  image_path = "../../gdrive/MyDrive/crowd-counting-data/images/green1.png"
  batch_size = 4
  log_para = 1000
  block_size = 32

args = Args()
args.model_path
predict(args)
```

Original image:

Figure 3: Original drone image

SASNet image

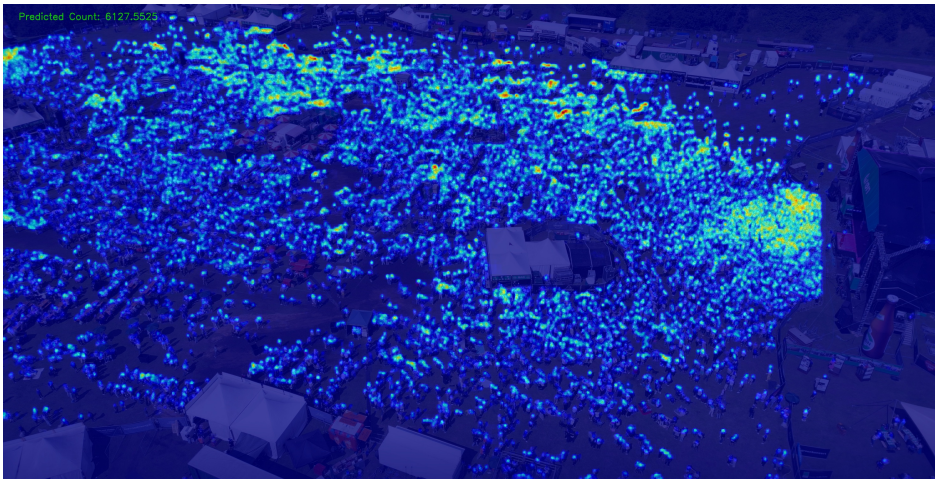


Figure 4: Overlayed image

Validation and Verification

Discussion

Conclusion

Perspective

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

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

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.

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

Sam, D. B., Agarwalla, A., Joseph, J., Sindagi, V. A., Babu, R. V., & Patel, V. M. (2020). Completely self-supervised crowd counting via distribution matching.

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)*.

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)*.