International Journal of User-System Interaction 12 (2) 2019, 67-82 © MatrixRom

Counting People in Crowded Places using Convolutional Neural Networks

Eduard Cojocea[1,2], Traian Rebedea[1,2]

[1]Open Gov SRL
95 Blvd. Alexandru Ioan Cuza, Bucharest, Romania
[2]University Politehnica of Bucharest
313 Splaiul Independentei, Bucharest, Romania
E-mail: iedi.cojocea@gmail.com, traian.rebedea@cs.pub.ro

Abstract. Security cameras have been around for decades and they continue to grow in number and capabilities as time passes. These days, in addition to the security nature of these cameras, they can be also used to obtain relevant information about the scene. Counting people and classifying them by gender, age, weight, height, or other attributes are very useful for marketing purposes and much more. Detecting anomalous events, like a suspicious luggage left unattended in a crowd or a person falling on subway rails could improve security in many spaces that are frequented by large masses of people. Extracting such data from video streams can have many more uses, some of which are not even imagined yet. In this paper we present how object recognition, tracking and classification methods can be used in conjunction to offer a solution for the problems described above.

Keywords: people counting, people profiling, deep learning, computer vision, object recognition.

1. Introduction

In this paper is described a solution for a system that can reliably extract statistics, in real time, from video streams of crowded indoor places such as malls, markets, subway stations, concert halls and so on and make them available to the user through a custom Graphic User Interface (GUI). These statistics are to be used for marketing purposes, serving as valid estimations regarding the size of the crowd and its gender, age, weight and height distributions. Obtaining low error estimations of the characteristics described above is the main goal of the system, with optional goals such as anomaly detection, which could be used for security purposes.

It is important to mention the fact that the system described in this paper is aimed to be implemented in real life scenarios. For this, three scenarios are explored, which involve significant differences in both hardware architecture and software.

The first scenario uses a client-server architecture, where the video stream is collected from a camera and is sent as a whole to a server to be processed. This involves using the great computing power of one or multiple servers for running the algorithms that detect persons, track them and extract profiling features (e.g. age, gender) in order to classify people in specific groups.

The second scenario consists of running the algorithms on an embedded device directly connected to the camera. This way, the latency and data traffic involving the first scenario are eliminated, but lower computing power is available.

The third scenario is a hybrid between the first and second scenario, involving processing the less computing intensive parts on an embedded device and sending the rest to a server in order to be processed remotely on a more powerful machine.

All three scenarios involve a further step of sending the extracted data to an application which will display the extracted information in real time to the user.

The tradeoff between running on a server with high computational power available and an embedded device is determined by the smaller and smaller gains obtained by using more and more computing power or computing intensive algorithms. It is still a matter of research if the increase in performance of some algorithms is really necessary for practical purposes. It may be the case that an algorithm with poorer performance will be preferred due to the fact that it requires significantly less computational power.

For this system to be functional, multiple algorithms are necessary, mostly in a pipeline. Firstly, an object detection algorithm is required in order to classify and localize objects in the video's frames. Secondly, an object tracker will use the objects' class and location in order to track them, which means to uniquely identify them in successive frames. Thirdly, multiple classifiers will use the uniquely identified objects in order to extract further information from them, placing these objects in subclasses. If the objects considered are humans, they will be placed in subclasses such as gender, age, weight, height and so on. Counting the objects can be done by relying on the tracker, which means counting the new ids from a certain moment of time. Alternatively, counting the objects can be done independently of the tracker, relying solely on the object detection, though this approach might offer estimates that are less accurate than the first approach.

The GUI through which the user can interact with the data extracted offers a variety of tools for data manipulation and visualisation. They will allow the user to extract higher level insights from the raw statistics, such as trends, cycles and other repetitive phenomena regarding the flow of people, which will be very useful for marketing and other similar purposes.

2. Related work

In this section there will be presented methods and algorithms regarding object detection, object tracking, object counting, and various classifiers used for extracting attributes from the objects detected in videos.

2.1. Object detection

Sutskever, Hinton, & Krizhevsky [1] described an architecture that won the ImageNet Large Scale Visual Recognition Challenge 2012, popularizing Convolutional Neural Networks (CNN), which became the staple of Machine Learning for images and videos due to their usage as general feature extractors. Continuing this work, several solutions have been found for detecting, localizing and classifying objects in images.

Region Convolutional Neural Networks (RCNN) [2] use a selective search algorithm in order to extract 2000 candidate regions, that might contain objects. These regions are then cropped and warped into 227x227 pixels images, which are in turn used as inputs in a CNN having the AlexNet [1] architecture. This way, the output consists of 4096-dimensional feature vectors, which are fed into a Support Vector Machine (SVM) in order to classify the object. The feature vectors are also fed into a regressor, which predicts 4 offsets in order to increase the precision of the bounding box. This is necessary, since sometimes the objects are only partially situated in a region, hence needing localization adjustment. Unfortunately, this approach could not be implemented in real time, since it needed tens of seconds in order to process an image. This is due to the large number of regions passed through the CNN. Also, the algorithm generating these regions is static, meaning that no learning occurs. Another disadvantage is that it needs a large amount of time for training (~84 hours as mentioned in [3]).

Fast RCNN [3] has a similar approach to the RCNN method, but instead of generating the regions and feeding each and single one of them into the feature extractor, the input image is directly fed into the CNN, outputting features maps. Regions are then proposed from the features maps, using the same selective search algorithm. These are then warped into squares, which are shaped into a fixed size by the Region of Interest (RoI) pooling layer, feeding them into a Fully Connected layer. For classification it is used a softmax layer, while a regressor predicts the bounding box offsets similar to RCNN. This method is a lot faster than RCNN, taking down the processing time for a single image significantly (in our experiments, from tens of seconds to several seconds). The training time is also improved, being necessary 9.5 hours (as presented in [3]) The bottleneck of this approach is the process of proposing regions.

Faster RCNN [4] is an improved version of Fast RCNN, where the selective search algorithm for proposing regions is eliminated. Using this new approach, the network learns to generate region proposals. The image is fed into the feature extractor, outputting features maps, which are further used as inputs into a Region Proposal Network (RPN). The proposed regions are then reshaped using a Region of Interest pooling layer and fed into a classifier and a regressor, similar to the previous 2 approaches. The great achievement made by this approach is that it can be used in real time, taking less than a second (a few hundred milliseconds) to process an image.

Mask RCNN [5] extends Faster RCNN by adding a predicting branch for objects masks specific to a class. This branch is added in parallel to the classifier and the regressor. Another change is the replacement of RoI pooling layer with a RoIAlign layer, which uses bilinear interpolation in order to obtain the exact location of the features at each subwindow.

You Only Look Once (YOLO) [6] is a totally different approach from those described earlier. It uses a single CNN for predicting both the offsets to the bounding boxes and the class probabilities for them. The input image is split into an SxS grid. For each grid are assigned m bounding boxes. For each bounding box, the network predicts a class probability and the offset values. If the probability is above a threshold, the bounding box is selected and used to locate the object in the image. This approach is much faster than previous ones, being able to run at 45 frames per second. The biggest drawback of this approach consists of the spatial constraints that don't allow too many small objects to be detected in a cell of the SxS grid. So this approach struggles to detect small objects, such as a flock of birds.

YOLOv2 [7] is an improved version of YOLO, implementing the following changes: batch normalization on all convolutional layers, convolutions with anchor boxes, direct location prediction, fine grained features, multi-scale training. These incremental changes made possible significant improvements in mean Average Precision (mAP).

YOLOv3 [8] is an improved version of YOLOv2, making use of a deeper architecture, with 106 fully convolutional layers. Unlike the first two versions, YOLOv3 implements residual blocks, skip connections and upsampling, which are usually present in state of the art algorithms. Other improvements include making predictions at three scales, better detecting smaller objects, increase in the number of bounding boxes per image and changes in the loss function. All three approaches, YOLO, YOLOv2 and YOLOv3 have variations both in architecture and in approach that allow a tradeoff between mAP and inference time. This is very useful in practice, especially in cases where there is not a lot of computer power available, but a real time inference is desired.

2.2 Object tracking

Object tracking is the ability of a model to uniquely identify one or more objects in successive frames. There are many solutions for tracking a single object, such as GOTURN [9] and MILTrack [10], but in the scope of this paper it is needed a multi-object tracker. Also, an online tracker, as opposed to a batch tracker, is needed for being able to run the system in real time.

Online Multi-Object Tracking by Decision Making [11] proposes an approach that formulates the multiple object tracking as a decision making problem. Thus, it models the lifetime of a tracked object using a Markov Decision Process (MDP), having 4 states for an object: Active, which is the state in which an object is detected; Tracked, which is the state where a true positive should transition; Lost, which is the state where the object is occluded or out of the scene; Inactive, which is the state where an object has been in the Lost state for long enough.

Simple Online and Realtime Tracking (SORT) [12] proposes a model that relies on the good performance of an object detector and uses a combination of simple techniques such as Kalman Filter and the Hungarian algorithm for the tracking. This approach is very fast, being able to easily run in real time, while also being online, meaning that only the frames until the current frame are known.

Simple Online and Realtime Tracking with a Deep Association Metric (DeepSORT) [13] builds upon the model of SORT by adding the appearance information about the objects tracked. This is done by pre-training a network on a large person dataset, which learns a deep association metric. This is very useful for objects that are occluded for more time or disappear out of the scene for longer. This metric also helps limiting the ID switching between objects that overlap or occlude one another.

3. System description

In this section will be described the structure and the first draft version of the proposed system.

Successful attempts have been made in order to use an embedded device dedicated for Machine Learning algorithms, such as Movidius [14] and Jetson [15], in conjunction with a Raspberry Pi. Moreover, initial promising results have been obtained by using a series of neural networks for object detection, custom made to be run on such devices, called Mobilenets [16]. However, the focus was shifted from this approach because establishing what performance could be achieved using the computational power of two GeForce GTX 1080 Ti GPUs had a higher priority. Thus, after having a working model from start to end, the effort could be redirected into optimizing and adapting the model for embedded devices.

In this system, YOLOv3 [8] is used as the preferred object detector, due to its great balance between performance and frames processed per second. Also, it is handy to integrate it code-wise. It was able to process 600x600 pixels images in ~0.02 seconds, which is around 50 frames per second, being reliable to run in real time. Also, shallower YOLO architectures have been tested.

For tracking, DeepSORT [13] has been the best solution to date for the problem at hand. In scenes expected for the system, such as entrances to shopping malls, there are many occlusions between people. In this particular case, DeepSORT is very useful due to its deep association metric, which allows for occlusions to last a few seconds. Also, the occlusions are being limited thanks to the flexibility of the camera placement. Thus, a camera placed higher eliminates some of the occlusions and makes other occlusions to be partial. On the other hand, by doing this, the bounding boxes with persons in the scene will have lower quality due to distance and angle.

DeepSORT extracts a deep association metric, consisting of a 128-dimensional features vector, which is projected on the unit hypersphere. This way, the similarity in appearance between two objects is determined by using the cosine distance between the two resulting 128-dimensional points on the unit hypersphere. If the value is below a certain threshold, then it is considered that the two objects are the one and the same.

In order to increase the tracking performance of the particular case of a high placed camera in an entrance to a mall, a grid search has been made over 3 important parameters of DeepSORT. The first represents the cosine threshold for the similarity in appearance of two objects. The second is the age of a tracked object, which means the number of frames that an object is kept after it disappeared from the scene or it has been occluded. This way, if the object is detected again after less than this number of frames have passed, it is recognized with its ID and not treated as a new object. The third parameter is the initial number of frames a new object has to be consistently tracked before it counts as a new object. This is important for eliminating stray objects or false object detections, which appear for only one or a few frames. For this grid search, there have been used a few hand annotated videos with real life scenarios. The optimal values found were 0.3 for the cosine threshold, 100 for the age and 4 for the initial frames.

Having the tracker working, counting the people in the video stream is a trivial task, equivalent to counting the new IDs registered by the tracking algorithm.

4. Architecture

In this section there will be described the flow of data from the camera/video file straight to the GUI application which displays the results.

In Figure 1 is shown the general architecture of the proposed system, which has three parts: Data Acquisition module, Computer Vision module, Application module.

4.1 Data acquisition module

This module is responsible with acquiring the video streams that are to be processed. This usually means using a camera placed at the desired scene, which can send the frames through a network protocol, such as Real Time Streaming Protocol (RTSP). Cameras usually support RTSP, making it easy for other devices to access the video stream through Ethernet of Wi-Fi. The streams are usually compressed using codecs, H.264 and its successor H.265 being very popular and having high performance in compressing videos without reducing the quality. Another option is to use only a sensor in order to get the raw frames. This would be more useful for embedded devices, where the sensor could be directly connected to the devices, bypassing any protocols. In Figure 1, it is shown that video streams could be accessed from video files as well. This is useful for test, historical runs or buffering videos in case of delays caused by the Computer Vision algorithms.

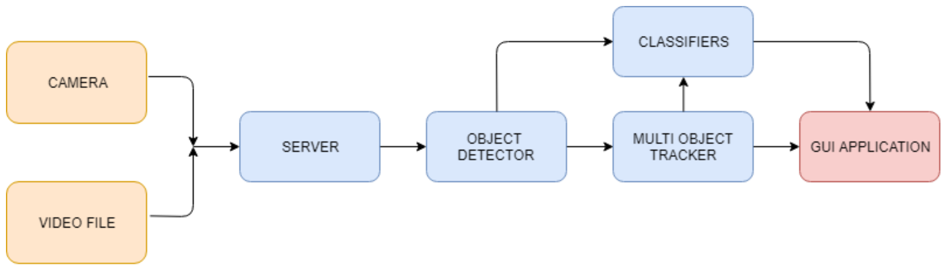

Figure 1 – The architecture of the proposed system

4.2 Computer vision module

This module is responsible with processing the video streams, regardless of their origin.

Whenever the system runs, it receives frames from a RTSP connection, a serial connection from a directly connected device or a video file. After each frame is received and the raw pixels are decoded, they are sent as input to the Object Detector, which outputs a list of objects, their centers, widths and heights. Then, the frames and the objects recognized within, together with their coordinates are used as input for the Tracking algorithm. Note that the relevant objects are persons. Whenever a person is considered new and tracked, the Tracking algorithm sends a HTTP request to the GUI application, containing the number of new persons. Besides the HTTP request, the Tracking algorithm, along with the Object Detector, sends information to a number of classifiers. The information needed as input consists of the bounding boxes of new persons detected. Then, these images will be classified and the results sent to the GUI application. The results will consist in a number of males and females, number of children, teens, adults and elders. Categories for weight and height will be set as well.

4.3 Application module

This module is responsible for taking the results from the Computer Vision module and displaying them using a GUI containing charts, graphics and other visual representations for data. Also, this module is responsible for offering the user various filters and time buckets such that the data received could be visualized in as many useful ways as possible. In Figure 2 is presented the GUI prototype implemented. It contains 4 main screens, each representing the data in different formats. The first screen is a timeseries chart, where the number of persons detected are ploted against time. The second screen presents a pie chart with the data. The third screen is a heat map of people detected and the fourth screen is a bar chart of the same data. Each screen has tools that allow the user to filter the data by gender, age and type of detection: "snap" means people detected in frames, "track in" means people detected that are coming in the scene, "track out" means people detected leaving the scene. The data used in Figure 2 is extracted from videos saved from live cameras in a shopping mall by the computer vision module. It must be mentioned here, that while the number of people detected and counted are a direct result of the models used, the classification by gender and age is done randomly. This is done in order to test all of the GUI's tools.

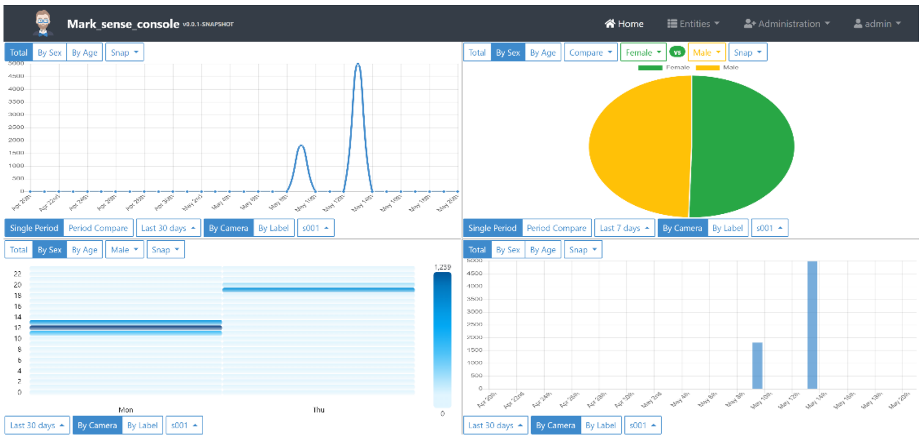

Figure 2 – The GUI prototype for interaction with the system

5. Future additions

In this section there will be described the components that will be added to the system in order to achieve its goal. This includes the classifiers for each person and some anomaly detection models.

5.1 Object detection and tracking

At the time of the writing of this paper, the system has a fully functional object detector and tracker. It is planned that in the next few months to improve the performance of these models. Also, it is desirable to try running smaller versions of these models on embedded devices, in order to investigate the validity of such an approach.

5.2 Classifiers

For the time being, no classifier has been implemented yet. But extensive research has been conducted in this field, resulting in a roadmap of steps that need to be followed in order to achieve a fully functional system.

Wu et al. [17] present multiple models that are able to classify persons by gender. It presents both appearance and non-appearance methods, though the latter is not in the scope of the system, since the only input consists of video streams from fixed cameras. Nithyashri & Kulanthaivel [18] propose a method for age classification into four categories: children (0-12 years), teenager (13-18), adult (19-59 years) and elders (60+ years). Rativa, Fernandes & Roque [19] describe a method based on anthropometry for the estimation of height and weight, which can be adapted for this system. The solutions described up to this point have encouraging results, which makes them perfect candidates to be integrated into the system pipeline.

Also, for the classifiers needed, custom networks can be trained, using as foundation one of the many CNN architectures that have proven to have high performance, such as AlexNet [1], ZFNet [20], VGGNet [21], GoogLeNet [22], ResNet [23].

6. Encountered issues

In this section a special focus will be given to the most challenging issues encountered during the development phase. Also of intereset are the expected issues regarding the performance of the system.

Some of the issues met were regarding the object detector. One issue is the poor detection of objects far from the camera, when they appear smaller in relative size. This means that the detection area had to be reduced. Also, due to the grid nature of YOLOv3 [8], when many objects from the same class are in a vicinity to one another, some false positives are detected, considering the whole group of objects as a single object of the same class. This anomalous behaviour required a special filtering of the detections.

When it comes to the tracker, DeepSORT [13], its performance varies considerably, and is correlated with the three parameters described in section 3, which required extensive tweaking for the right parameters for the specific case at hand. Also, the performance of the tracker drops significantly when the number of objects increases by a big factor. Basically, in tightly packed crowds, the tracker has the lowest performance. This is to be expected, since there are a lot more objects to track and most objects have many occlusions, some of them being occluded most of the time or for extended periods. One thing that is noteworthy is the fact that the tracked error is consistent with the object density in the scene. This means that the tracker could be adjusted, with the help of the detector, by correcting the number of objects being tracked. Also, a statistical approach for estimating the number of persons in a video stream is worth exploring.

7. Results

At the time of the writing of this paper, the only results available from the implementation of the system are regarding people counting. We will present them in this section, though an accurate measurement is very difficult and time consuming. The numbers presented regarding the ground truth are estimations, due to the nature of the video files used and their length. In other words, the exact number of unique persons in the videos is hard to be computed even by humans. The system is tested this way and not using a largely used annotated dataset, since it has a particular purpose. Thus, videos from real life scenarios are used.When it comes to the object detector's performance, it is worth noting that the detector was tested in real life scenarios, with a lot of noise and far from ideal camera placement. The results obtained using YOLOv3 are presented in Table 1.

Table 1 – The results obtained by the proposed method on real life data

Method	AP	AP_{50}	AP_{75}
Faster R-CNN+++	34.9	55.7	37.4
YOLOv2	21.6	44.0	19.2
RetinaNet	40.8	61.1	44.1
DSSD513	33.2	53.3	35.2
YOLOv3 – original	33.0	57.9	34.4
YOLOv3 – custom	27.9	53.2	30.1

As it can be seen in Table 1, the model on this specific set of data (e.g. fixed cameras at mall entrances, a few meters above the ground) behaves slightly worse than its original, but this is to be expected. However for the scope of this paper (e.g. counting people), the performance achieved is acceptable, while further improvements considering the object detector's performance can be added subsequently, or even another object detector can be used in the place of the current one. The videos used for measuring the performance where annotated by the authors, containing real life scenarios from shopping malls, ranging in length from 15 seconds to 300 seconds. The people density in the videos varies from a couple of persons to tens of persons at a time in a frame. In order to improve the performance of the overall system, the focus should be shifted towards improving the object tracker.

There are two performance metrics that concern the algorithm: the difference between the ground truth and the prediction and the number of frames processed per second. Both of these metrics vary according to the scene. This is to be expected, taking into account the two main components of the current system: an object detector and an object tracker. Both components have better performances, both in quality and frames per second, when fewer objects are in the scene. Also, their performance is increased by a lower number of occlusions. But as the number of objects and occlusions in a scene increase, their performance drops and can process fewer frames per second.

Table 2 – Results of the system on two relevant videos

Video	Predicted	Ground Truth	Min fps	Max fps	Average fps
First	74	68	9	30	21.75
Second	5624	2157	3	20	7.76

The results in Table 2 are obtained from two relevant videos. The first one represents a short video, 15 minutes in duration, with a relatively small number of persons and occlusions. It depicts a scene from the entrance to cinema halls. The second video represents a longer video, 60 minutes in duration, with a higher number of persons and occlusions. It depicts a scene from the cinema bar, where people pass by or stand in line.

It is visible that the error increases greatly in the second video. But it is worth noting that the error is directly proportional to the number of persons and occlusions, which will allow in the future for corrections to be made heuristically in post processing.

In Figure 3 is presented a frame from the second video. The blue bounding boxes represent the object detector's prediction, while the white bounding boxes represent the tracker's prediction of the future position for each object detection at the previous frame. The numbers shown in green are each person's unique ID. It can be seen that many persons are not detected due to occlusions or their small relative size.

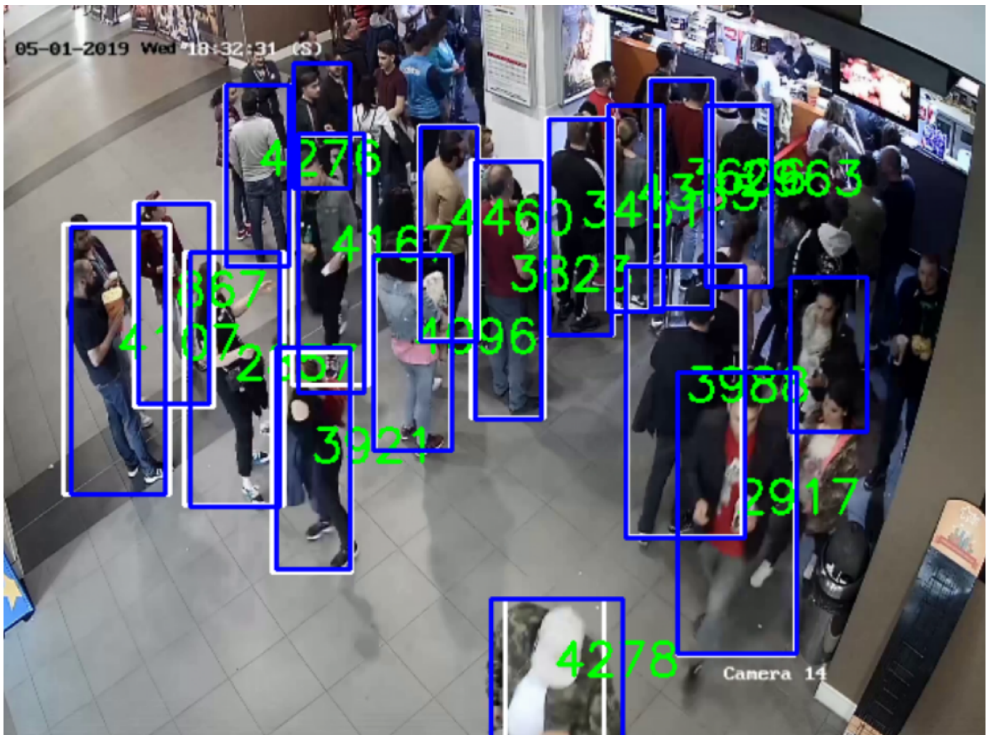

Figure 3 – Capture of a frame from the second video as processed by the proposed system

The high number of occlusions and small relative sizes of persons induce errors both in the object detector, as well as in the tracker (since the tracker is based on the object detector), which induces even bigger errors in the overall system.

The performance of the GUI will be tested in the near future, with users in real scenarios, and upgraded accordingly, in order to ensure the novelty and usefulness of the system.

8. Conclusions

In this paper we have presented a system which is intended to collect video streams from indoor cameras or using prerecored videos, extract relevant information about the people detected in the videos, such as number of persons in a certain interval of time and their age, gender, weight, height etc. distributions. These data are then sent to a custom GUI for visualisation and filtering to empower the users to easily extract statistics about the customers visiting crowded open spaces, such as shopping malls. In order to extract these data, the system uses an object detector [8] and an object tracker [13], which work in conjunction in order to estimate the number of persons. The system was tested on a dataset of real life videos, manually annoted for the scenarios considered in this paper (e.g. indoor malls).

The system presented in this paper is only partially implemented and still requires further improvements and optimization alongside implementing the classifiers for human profiling (e.g. age, gender). The current results are promising and the research and development process ahead is sure to achieve the expected results.

ACKNOWLEDGEMENTS

This research was funded by the MARKSENSE project "Real Time Analysis Platform For Persons Flows Based on Artificial Intelligence Algorithms and Intelligent Information Processing for Business and Government Environment", contract no. 124/13.10.2017, MySMIS 2014 code 119261.

References

1. Sutskever, I., Hinton, G. E., & Krizhevsky, A. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 1097-1105.
2. Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE

conference on computer vision and pattern recognition (pp. 580-587).

3. Girshick, R. (2015). Fast r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448).
4. Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems (pp. 91-99).
5. He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 2961-2969).
6. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).
7. Redmon, J., & Farhadi, A. (2017). YOLO9000: better, faster, stronger. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7263-7271).
8. Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.
9. Held, D., Thrun, S., & Savarese, S. (2016, October). Learning to track at 100 fps with deep regression networks. In European Conference on Computer Vision (pp. 749-765). Springer, Cham.
10. Babenko, B., Yang, M. H., & Belongie, S. (2009, June). Visual tracking with online multiple instance learning. In 2009 IEEE Conference on Computer Vision and Pattern Recognition (pp. 983-990). IEEE.
11. Xiang, Y., Alahi, A., & Savarese, S. (2015). Learning to track: Online multi-object tracking by decision making. In Proceedings of the IEEE international conference on computer vision (pp. 4705-4713).
12. Bewley, A., Ge, Z., Ott, L., Ramos, F., & Upcroft, B. (2016, September). Simple online and realtime tracking. In 2016 IEEE International Conference on Image Processing (ICIP) (pp. 3464-3468). IEEE.
13. Wojke, N., Bewley, A., & Paulus, D. (2017, September). Simple online and realtime tracking with a deep association metric. In 2017 IEEE International Conference on Image Processing (ICIP)(pp. 3645-3649). IEEE.
14. Intel Neural Compute Stick. https://www.movidius.com.
15. Nvidia Jetson TX2 Module. https://developer.nvidia.com/embedded/jetson-tx2.
16. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.
17. Wu, Y., Zhuang, Y., Long, X., Lin, F., & Xu, W. (2015). Human gender classification: A review. arXiv preprint arXiv:1507.05122.
18. Nithyashri, J., & Kulanthaivel, G. (2012, December). Classification of human age based on Neural Network using FG-NET Aging database and Wavelets. In 2012 Fourth International Conference on Advanced Computing (ICoAC) (pp. 1-5). IEEE.
19. Rativa, D., Fernandes, B. J., & Roque, A. (2018). Height and Weight Estimation From

Anthropometric Measurements Using Machine Learning Regressions. IEEE journal of translational engineering in health and medicine, 6, 1-9.

20. Zeiler, M. D., & Fergus, R. (2014, September). Visualizing and understanding convolutional networks. In European conference on computer vision (pp. 818-833). springer, Cham.
21. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.
22. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).
23. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).