In: 2023 IEEE Smart World Congress (SWC), 28-31 August 2023, Portsmouth, UK
DOI:10.1109/SWC57546.2023.10448959

Real-Time Vision-Based Violent Actions Detection Through CCTV Cameras With Pose Estimation

Paul Benoit, Marc Bresson, Yang Xing, *IEEE Member*, Weisi Guo, *IEEE Member*, and Antonios Tsourdos

***Abstract*— In large structures under video surveillance, or when a place is crowded, a CCTV operator cannot monitor hundreds of people on different video streams. This paper presents a proof of concept for a real-time vision-based system for detecting violent actions through CCTV cameras with pose estimation. The proposed system uses a combination of computer vision techniques, including pose estimation, object tracking, and a deep learning algorithm based on time-series features, to accurately identify violent actions in real-time. Our features are based on a fixed-size rolling window that computes the position of each person's limb along with their velocity. The proposed pipeline achieves a high accuracy rate of 92% with an overall latency of around 0.07 seconds per frame using a RTX 3060 Mobile GPU, making it a powerful tool for enhancing public safety and security. This system can be deployed in a wide range of scenarios, including public places, transportation hubs, and other critical infrastructure, to provide real-time alerts and facilitate rapid response in case of violent incidents.**

Keywords—Violence Detection, Pose Estimation, Near-Real Time, YOLOv8, Deep SORT

I. Introduction

Sudden violent behaviour can be hard to predict and usually happen very fast. For a security team, it is essential to be able to detect them as soon as possible for the safety purpose. In the United Kingdom the latest analysis suggests that over 5.2 million closed-circuit televisions (CCTV) are used for monitoring homes, businesses, or public locations [1]. In big structures such as airports, human surveillance is reaching its limits by becoming too costly and is subject to more failure considering the maximum number of CCTV an operator can manage at a time [2].

Faster detection would allow a prompt response from the closest security agent in the area, thus reducing the impact of these behaviours before the situation can escalate. This work proposes an integrated system based on computer vision to estimate and monitor the pose of persons identified in a CCTV feed, allowing the system to flag abnormal actions and track the person culprit. The system is not fully autonomous and expects an operator to confirm the detected event before dispatching security, but it would significantly increase the amount of CCTV an operator can handle.

In this study, the creation of features for the violence detection Artificial Intelligence (AI) model is based on already existing algorithms such as Deep SORT [3] for tracking and YOLOv8 [4] for pose estimation. The system can be divided into five subsystems: video extraction, pose estimation, tracking, classification, and notification. The main contributions of this paper are the following:

- Our paper presents a proof of concept of a time-distributed framework that analyses a sequence of poses over time to accurately identify and classify violent movements that we implemented and tested in real life conditions.
- We also propose an efficient system designed to process and analyse video stream data in near-real time using a low-latency architecture capable of handling video stream at a constant 10 frame per second (FPS) which is enough for its purpose.

All the codes are available on our GitHub [5] and can be found on pypi.org under the name "tactus-live" [6].

This study is organised as follows: section II is a review of the current literature on violent action detection. Section III explains the approach employed to establish the framework of the system. Section IV focuses on the specific design of the system, including the dataset, posture estimation and tracking, the approach taken to process data, the training and testing method, and Saab real-life testing. The results are presented in section V and then discussed in section VI along with the potential improvements for the system. Finally, section VII concludes the study.

II. Literature Review

Violent behaviour detection is a widely researched topic, so we started this project with an extensive literature review on the subject. Many different approaches are using various techniques such as the Internet of Things, AI, and also humans. To narrow the search and be relevant with respect to this paper, we will focus on automated Machine Learning and Deep Learning approaches.

A. Violence Detection

Thanks to the growth of micro-electro-mechanical systems. Sensors are getting smaller, cheaper, and more performant than before, allowing them to capture a wide range of physiological signals but also physical data such as movement and sound. The main issue with this method is how intrusive these solutions can be. There is a need for a wearable on every person for it to be effective. Tchuente, Baddour and Lemaire [7] discussed the feasibility of a solution using Smartwatches to get the data and using ML classifiers such as k-Nearest Neighbour with very high accuracy results. One other limitation of this method is that the range of detected action is limited.

*Research supported by Cranfield University.
The authors are with the School of Aerospace, Transport, and Manufacturing, at Cranfield University.
Email: {paul.benoit.927, marc.bresson.359, yang.x, weisi.guo, a.tsourdos}@cranfield.ac.uk.

© 2024 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works

A more common approach is to use video as the input data of the system. One way to identify violence is to create a binary classifier detecting violence or not, like in the paper from Vijeikis, Raudonis and Dervinis [8] that uses MobileNet V2 and LSTM. This approach reduces the amount of information in the prediction because there is no information about what type of violence has been detected, but it also increases the overall accuracy of violence detection.

B. *Pose Estimation*

Another approach is to use pose estimation to generate features and then classify them. There are two main approaches to estimate the pose of people given an image:

1) Top-down approaches where you first identify all the persons on the frame and then all the landmarks are computed for each person, Mask R-CNN is a commonly used Top-down approach. He, Gkioxari, Dollár and Girshick [9] discussed how to exploit Mask R-CNN to conduct a multi-person detection and pose estimation in one single framework for better accuracy and faster inference which is one of the issues of usual Top-down approaches when a lot of people are in the frame.

2) Bottom-up approaches identify all the landmarks directly creating a heatmap and then generating individuals based on all the features detected. It makes the approach less complex computationally wise and faster even in a crowded environment, but often less accurate than Top-down methods. In 2013, Yang and Ramanan [10] discussed a Flexible Partial Mixture model different from the traditional classification that takes the bending and shortening of the limbs into account by dividing limbs into small parts. Later in 2016, Cao, Simon, Wei, and Sheikh [11] introduced an alternative bottom-up parsing method known as Part Affinity Fields. This approach creates a confidence map of body parts and a vector field of part affinities to represent the probability of a particular body part at each pixel location, encoding the limbs and their connections to obtain 2D pose images of multiple individuals. Both methods have been discussed in-depth in [12].

C. *Limitation of the Current Methods*

In the field of violence detection, there is a lack of standardized datasets, and this makes the methods reliable only in certain configurations or angles. Our system does not solve this problem since it is a proof of concept but with the global architecture, it would be feasible to increase the number of datasets used to make the algorithm more versatile. Another thing is the high number of False Positive (FP) alarm rates in opposition to the precision which we optimised during the training phase. Sometimes even if the model used is great, CCTV video can have low resolution making it very hard to identify anything, this is where "super-resolution" AI model techniques such as [13] could be applied using Kernel estimation and Noise Injection. One major issue that the system solves is occlusion, if a place is very crowded then a person might stop being detected for a few frames. Existing solution discussed by Cheng et al. [14] allows occlusion solving for 3D pose estimation. We used another technique since our system is only 2D: it tries to track people based on the movement in the previous frames to be as accurate as possible and solve the occlusion issue.

III. Methodology

The goal being to design a real-time vision-based violent behaviour detector and tracker, we decided that our study would act as a proof of concept starting with only 4 simple actions detected: punching, pushing, kicking and the neutral class which represents every other action. For this study, we decided to classify every individual in the frame one by one and handle occlusion using Deep SORT functionalities. The features are composed of multiple normalized metrics generated from a rolling window allowing us to use a time-series model. We used a fully connected Multilayer Perceptron (MLP) AI network [15] to predict the ongoing action and then send it through an Apache Kafka Server structure to manage events logs [16].

One of the main goals of this study was to create a scalable solution that could cover big CCTV networks such as in a smart airport. Thus, the project is divided into multiple building blocks that we can instantiate multiple times for each camera. It starts with the video input sent to the live processing pipeline through RTSP protocol which is not the most recent protocol but still vastly used for CCTV camera streaming with around 2-second latency [17], it was also one of the protocols used by DARTeC live camera that we used during the testing. The video stream is split into frames at a set frame rate, processed into features, then saved into a fixed-size rolling window, and finally fed to the MLP classifier. Fig. 1 shows a broad view of the system architecture as follows:

1) Get the camera feed and extract frame by frame for processing.

2) Pose estimation and tracking.

3) Rolling window update and Feature extraction, followed by classification with an MLP.

4) Intercept video stream to add visuals (bounding boxes and classification results) and send notification through Apache Kafka Server.

The video output has been designed to show the bounding box of tracked individuals with a different colour code depending on the situation: red if they did a violent action at some point in the footage, green if nothing happened and grey if they are detected but not tracked for long enough to start predicting.

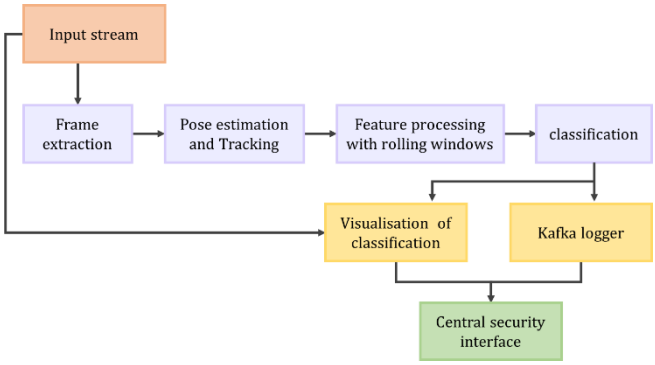

Fig. 1. Global architecture of the proposed system.

IV. System Design

A. Dataset

Our unique source of data comes from the UT-Interaction dataset [18]. It contains 120 labelled videos with 2 people doing interaction with each other (punching, kicking, pushing, pointing, hugging and handshaking). For this dataset, we manually labelled every video frame by frame because they were not correctly labelled for our needs. Each person in the video has a label per frame. We defined the start/end frame of each action and then set the rest to neutral. Two labelling strategies were used and tested changing how strict we were about the start/end of each action. With this technique, the dataset was unbalanced which could induce a bias in the model, but it was solved during the training phase.

B. Frame Extraction

To ensure the correct functioning of our algorithm, we must make sure that the processing frame rate is constant no matter the stream frame rate. For that, we created a custom interface of the popular computer vision library OpenCV-python that allows us to subsample any stream to a desired frame rate target. For this project, we picked a 10-fps frame rate to have a great compromise between efficiency and accuracy. If we consider a punch (an arm has less inertia than a leg, thus accelerating faster), its average speed among boxers is about 32km.h-1 [19]. Given the significant acceleration and deceleration time, the average boxer's arm can fully extend in 3 frames (0.2 seconds). Thus, 10 fps are enough to see the fastest action happening in a crowd. It must be noted that we could raise this number to 30 fps but that would require retraining the model at the correct frame rate and also induce more computational power and more memory for no significant gain of performance.

C. Pose Estimation

For the pose estimation, we used the 8th (and latest) version of YOLO [4] with the "YOLOv8m-pose" pre-trained model. It takes as an input a 640 by 640 image, which we achieved by padding and resizing the frame. For every frame, we extract every person's limb position along with the person's bounding box. The results are stored inside a custom "skeleton" object that stores the bounding box, the keypoints and the confidence score. Furthermore, it allows us to remove the « head keypoints » (ears, nose, eyes) since they are not articulated one to another thus redundant, and replace them with a single head keypoint as shown in Fig. 2.

YOLOv8 comes with multiple pre-trained pose models of different sizes. Because the pose estimation step is the bottleneck in our pipeline, the usage of one model over another offers great granularity to modulate the latency of the pipeline as shown in Fig. 3.

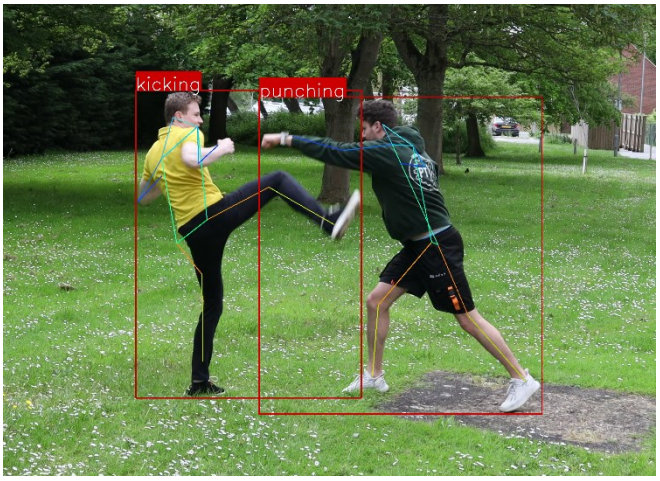

Fig. 2. Skeleton of the two authors of this paper, estimated with YOLOv8.

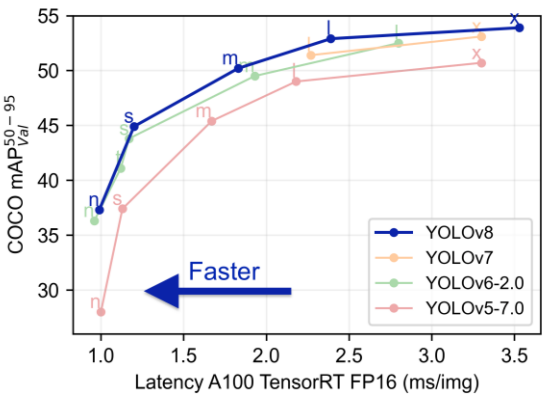

Fig. 3. Speed vs performance of YOLO [4].

D. Tracking

For the training, each video only contained 2 people or less. To avoid uncertainty induced by a complex Re-Identification (ReID) algorithm, we used a custom one that relies on the fact that the two persons never switch sides.

However, we used Deep SORT for the inference phase. It is a ReID algorithm derived from the SORT algorithm [20] that uses Kalman filters. We mapped the unique id number that we get from the tracking output to a rolling window that contains the history of a person's skeletons. The size of the rolling window is a parameter that we optimised during the training phase. Deep SORT can also maintain a track even if the skeleton has not been detected by YOLOv8. In such a case, we duplicate the last known "skeleton" of the rolling window. If a track has not received any updates on 20 frames in a row (equivalent to 2 seconds at 10 fps), it is discarded because it becomes too unreliable afterward.

The unique ID also allows us to track someone who had a violent behaviour at one point in the video or the stream, and flag him on all the next frames.

E. Data Processing

From the rolling window of a single tracked individual, we can extract and generate multiple features:

- The position of each keypoint relative to the neck, normalized with the height of the skeleton, is defined as the distance between the neck and the ankles.
- The angles of body joints like between the front leg and the thigh, between the front arm and the arm, or between the shoulder line and the arm. It came out that these features were not helping our model, so we ended up not using them.
- The relative speed of each keypoints compared to their previous position.

F. Multithreading and Multiprocessing

The time spent processing one frame is mostly spent executing YOLO (~40% of the frame processing time) followed by Deep SORT (~30% of the frame processing time). With multiprocessing, we can instantiate multiple models at a time and potentially reduce the number of frames skipped thanks to parallel frame processing. It could not be achieved using multithreading because of the Python per-process Global Interpreter Lock (GIL) which avoids concurring access of memory [21]. It is important to note that all the AI models for one process (YOLOv8, Deep SORT, Classifier) weigh 1.3GB in the GPU RAM, and so multiple instances of the pipeline could be loaded on the same GPU because the GIL is not shared between processes.

V. Model Development and Evaluation

A. Deep Learning Classifier Design

Before training plenty of models, we conducted a first baseline model performance analysis to test which algorithm would give the best results without in-depth tuning and Fig. 4 shows that MLP got the highest results. For the training, we decided to do a grid search which is a hyperparameter tuning method where you test all the different combinations from a grid of parameters. To do that we listed which MLP hyperparameters, data augmentation and feature processing parameters we would optimise during the training.

B. Model Training

As mentioned, multiple models were trained using a grid search on an NVIDIA A100 GPU over 300 epochs and categorical cross-entropy was used as a loss function. The grid search was done using 3 grids, with the following parameters:

1) Data augmentation grid: add noise, flip skeleton horizontally, flip skeleton on itself, rotate (X, Y, Z axis), scale.

2) Data processing grid: rolling window size, number of angles.

3) Classifier hyperparameters grid: activation function, hidden layer architecture (layers, dropout)

On top of that, different strategies were used to:

1) Balance the data: using Random UnderSampling (RUS), Synthetic OverSampling Minority (SMOTE) or by reducing the number of neutral videos from the original dataset.

2) Label the data: changing our criteria on when the action starts and ends.

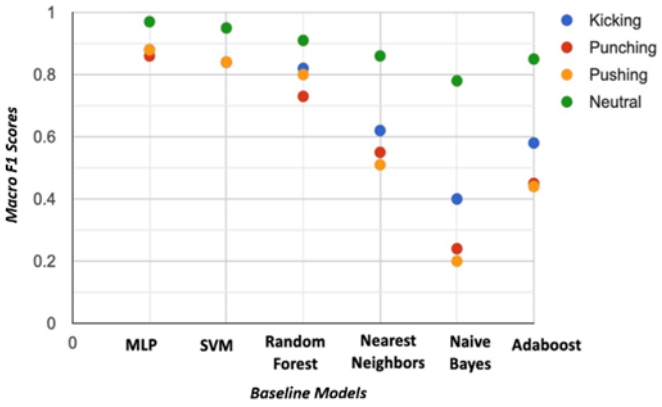

Fig. 4. Baseline model performance on F1 Scores

C. Model Evaluation

As mentioned in the previous section, many models were generated. To evaluate them we searched to maximize F1-score (4) because it is a balance between FP, False Negative (FN), True Positive (TP) and True Negative (TN). The model must have a low amount of FP to not flood the human confirmation task, but it also needs to not miss violent actions, which translates into reducing FN. Since we have a multiclass detector and F1 Score is a binary metric, we had to compute one F1 score per class and average them to get the final F1 score, which is called "Macro F1 score".

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (4)$$

At the end of the training, even if the results were similar, the stricter labelling strategy (regarding the beginning/end of action), combined with the use of RUS for balancing data gave slightly better results than other combinations. The final architecture is a 4-layers MLP with the following layer widths: 468/64/32/16/4. We used LeakyReLu as the activation function of the hidden layers, and Softmax for the output layer.

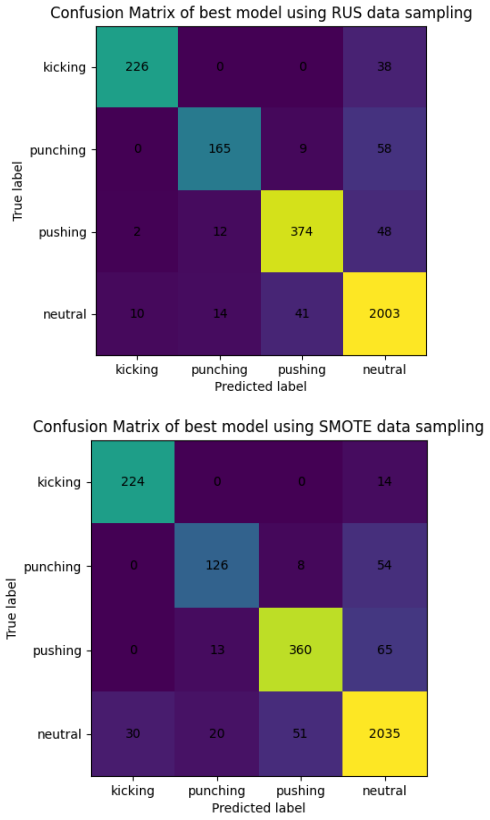

Fig. 5. Confusion Matrix using RUS and SMOTE

Confusion Matrix is used to evaluate the multi-class classification performance, as shown in Fig. 5 and Table I. We can draw a few conclusions. For example, no matter the model, there seems to be a bias towards the neutral class. It is not surprising considering there could be plenty of similar actions such as pointing and punching skeleton-wise. The results are encouraging because violent actions are rarely misclassified between themselves even if some neutral actions are similar to violent ones, e.g., pointing and punching.

TABLE I. CONFUSION MATRIX SUMMARY

Model	*TP*	*TN*	*FP*	*FN*
RUS[a.]	765	2003	144	88
SMOTE[b.]	710	2035	133	122

a. RUS = Best model with RUS during grid search, [b.] SMOTE = Best model with SMOTE during grid search.

TABLE II. BEST MODELS RESULTS

Model	*Train Accuracy (%)*	*Test Accuracy (%)*	*Macro Recall*	*Macro Precision*	*Macro F1 Score*
RUS	0.995	0.92	0.897	0.842	0.868
SMOTE	0.995	0.915	0.853	0.842	0.848

As we can see in Table II., the model is capable of 92% accuracy and proves its robustness to identify the currently flagged actions. Despite the rolling window, we still had a few actions wrongfully flagged on one frame, so we decided to notify only if someone was flagged on multiple consecutive frames to reduce the number of FP.

Overfitting is occurring in the model as we can see from the difference between train and test accuracy. The overfitting could be limited by increasing the different actions counted as Neutral (currently only handshake, hug and pointing). One of the major issues is the amount of FN and FP as we can see in Table I., this could lead to operators missing major incidents and possibly wasting time on errors. We also need to consider the fact that the model was given testing data in the same context as the training, and it could have worse results with an unusual angle and camera.

The whole system is also complying with the near real-time expectations with a 2-second delay between the action happening live and the alarm being sent for a normal architecture. It is mainly due to the slowness of the network infrastructure and the RTSP protocol; this AI pipeline is only accountable for ~ 0.07 seconds of delay using a RTX 3060 Mobile GPU.

D. Saab SAFE Real-life implementation

To test our system further, we implemented it in collaboration with Saab SAFE Client – an open-integration software platform for mission-critical operations used in command-and-control rooms [22]. This client is made by the company Saab AB, and we were able to use it and adjust our layout on the client screen that the security operator would see. We could not completely exploit this powerful tool and had many ideas for future improvement. The client was set up on a Virtual Machine instance that was able to receive the Kafka log packages thanks to a special gateway designed by Saab and DARTeC. To test it, we connected the client to DARTeC CCTV camera and went there to simulate violent behaviour. The test was successful, and we managed to identify violent actions and send alarms to the SAFE client. Compared to our local testing, the client added a slight delay which is small enough to keep the near real-time requirement.

VI. DISCUSSION AND FUTURE WORKS

The system was able to provide solid results while still having lots of room for improvement. The first thing is that we only tried 2D pose estimate but 3D (although more complex to implement) would allow the classifier to get more precise data and to predict situations from many angles with fewer errors. The idea is not out of the picture but would need a redesign of the architecture since YOLOv8 cannot do 3D pose estimation.

Furthermore, YOLO has been trained with 640*640 images which forces us to resize and pad the frames we extract from a camera. It leads to a significant decrease in resolution and the pose estimation model can then miss the smallest skeletons. There are two solutions to counteract this issue: slice a frame into multiple 640*640 images and merge the prediction together; or train a custom YOLO model on 1920*1080 images – a common resolution and aspect ratio in CCTV.

The results of the model are satisfying, one big limitation was the dataset which only identifies 3 violent actions. It is limited and relatively small, even with data augmentations. Since we want to be able to detect as many actions as possible, we could use multiple classifiers identifying 3-5 actions since our MLP architecture is light and fast. We can assume that the Macro F1 score could be significantly improved by using more data from multiple datasets but increasing the number of classes identified could end up lowering the overall accuracy as a trade-off. Besides, many datasets of violence on CCTV exist but are not labelled by frame, and fusing multiple datasets to identify more actions would have been a time-consuming task. To avoid manual labelling, we looked at solutions combining gradient activity detection and Motion History Image followed by human validation [23]. Since we labelled manually, our system might be biased by the subjective labelling strategy that we used.

To improve the system even further we could use multiprocessing, but it is heavier in terms of RAM. Another solution might also become available with the new release of Python 3.12 allowing the use of a per-interpreter GIL which would sidestep GIL limitations and allow easier multicore programming using sub-interpreters [24]. Another future work is keeping gathering more data to train the model with real-life situations and get better performance as it is used with monthly updates to enhance it.

VII. Conclusion

In this work, we propose a working violent action detector live pipeline using pose detection and MLP classifier. We used a new architecture allowing us to generate features and store them in time series. Many data augmentation techniques and data processing techniques were explored to find the best combination. Results are encouraging as proof of concept and could be improved by tweaking the model architecture and more importantly improving the dataset. The system in its current state was successfully integrated into Saab SAFE software through a special gateway.

Acknowledgment

We thank Avishaek Deep, Pradnya Patil and Ruochen Yin for originally working with us on the project as part of Cranfield University group project. We also thank the Saab SAFE team for their support of the SAFE environment integration with particular thanks to Wikander Andreas and Henningsson Andreas. We authors also thank George Yazigi for his support in managing computing resources and the environment at Cranfield DARTeC.

References

[1] HR News, "Number of CCTV Cameras in the UK Reaches 5.2 Million.," 19 November 2020. [Online]. Available: https://hrnews.co.uk/number-of-cctv-cameras-in-the-uk-reaches-5-2-million/.

[2] H. M. Hodgetts, F. Vachon, C. Chamberland and S. Tremblay, "See No Evil: Cognitive Challenges of Security Surveillance and Monitoring.," Journal of Applied Research in Memory and Cognition, vol. 6, no. 3, pp. 230-243, 2017.

[3] N. Wojke, A. Bewley and D. Paulus, "Deep_sort: Simple Online Realtime Tracking with a Deep Association Metric," Github, 2017. [Online]. Available: https://github.com/nwojke/deep_sort.

[4] Ultralytics, "NEW - YOLOv8 in PyTorch > ONNX > CoreML > TFLite. n.d," Github, 2023. [Online]. Available: https://github.com/ultralytics/ultralytics.

[5] Cranfield-GPD3, "TACTUS - live" Github, 2023. [Online]. Available: https://github.com/Cranfield-GDP3/TACTUS-live.

[6] Marc Bresson, "tactus-live 1.1.0" Pypi.org, 2023 [Online]. Available: https://pypi.org/project/tactus-live/.

[7] F. Tchuente, N. Baddour and E. D. Lemaire, "Classification of Aggressive Movements Using Smartwatches," Sensors, vol. 20, no. 21, p. 6377, 2020.

[8] R. Vijeikis, V. Raudonis and G. Dervinis, "Efficient Violence Detection in Surveillance," Sensors, vol. 22, no. 6, p. 2216, 2022.

[9] K. He, G. Gkioxari, P. Dollár and R. Girshick, Mask R-CNN, 2017

[10] Y. Yang and D. Ramanan, "Articulated Human Detection with Flexible Mixtures of Parts.," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, pp. 2878-2890, 2013.

[11] Z. Cao, T. Simon, S.-E. Wei and Y. Sheikh, Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields, 2017.

[12] S. Jin, X. Ma, Z. Han, Y. Wu, W. Yang, W. Liu, C. Qian and W. Ouyang, Towards Multi-Person Pose Tracking : Bottom-up and Top-down Methods, 2017.

[13] X. Ji, Y. Cao, Y. Tai, C. Wang, J. Li and F. Huang, "Real-World Super-Resolution via Kernel Estimation and Noise Injection.," in IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2020, pp. 1914-1923.

[14] Y. Cheng, B. Yang, B. Wang, Y. Wending and R. Tan, "Occlusion-Aware Networks for 3D Human Pose Estimation in Video," in 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 723-732.

[15] J. Talukdar and B. Mehta, "Human Action Recognition System Using Good Features and Multilayer Perceptron Network.," in 2017 International Conference on Communication and Signal Processing (ICCSP), 2017.

[16] J. Kreps, "Kafka: A Distributed Messaging System for Log Processing.," 2011.

[17] R. Stone, "Video Streaming Protocols Compared | Mushroom Networks.," [Online]. Available: https://www.mushroomnetworks.com/blog/video-streaming-protocols-compared/.

[18] M. S. Ryoo and J. K. Aggarwal, "UT-Interaction Dataset, ICPR contest on Semantic Description of Human Activities (SDHA)," 2010. [Online]. Available: https://cvrc.ece.utexas.edu/SDHA2010/Human_Interaction.html.

[19] S. Writer, "What Is The Average Punching Speed? Punching Speed Analyzed," mmachannel, [Online]. Available: https://mmachannel.com/what-is-the-average-punching-speed-punching-speed-analyzed/.

[20] A. Bewley, Z. Ge, L. Ott, F. Ramos and B. Upcroft, "Simple Online and Realtime Tracking," 2016.

[21] S. Gross, "PEP 703 – Making the Global Interpreter Lock Optional in CPython.," Python Org., 9 January 2023. [Online]. Available: https://peps.python.org/pep-0703/.

[22] SAAB AB, "SAFE Airport," [Online]. Available: https://www.saab.com/products/safeairports.

[23] O. M. Sincan and H. Y. Keles, "Using Motion History Images with 3D Convolutional Networks in Isolated Sign Language Recognition," {IEEE} Access, vol. 10, pp. 18608-18618, 2022.

[24] S. Yegulalp, "Python 3.12: Faster, Leaner, More Future-Proof.," InfoWorld, 1 May 2023. [Online]. Available: https://www.infoworld.com/article/3694512/python-312-faster-leaner-more-future-proof.html

https://dspace.org/

Real-time vision-based violent actions detection through CCTV cameras with pose estimation

Benoit, Paul; Bresson, Marc; Xing, Yang; Guo, Weisi; Tsourdos, Antonios

2024-03-01
Institute of Electrical and Electronics Engineers (IEEE)
https://doi.org/10.1109/SWC57546.2023.10448959
Benoit P, Bresson M, Xing Y, et al., (2024) Real-time vision-based violent actions detection through CCTV cameras with pose estimation. In: 2023 IEEE Smart World Congress (SWC), 28-31 August 2023, Portsmouth, UK, pp. 844-849

Downloaded from a DSpace repository. https://dspace.org/