SCHOOL SURVEILLANCE SYSTEM FOR AN AUTONOMOUS LAND VEHICLE

Chia Mei Xi[1], Jessie Chin Kit Hey[1], Ng Pool Hua Kennard[2]
[1] Hwa Chong Institution (College), 661 Bukit Timah Rd, Singapore 269734
[2] DSO National Laboratories, 12 Science Park Drive, Singapore 118225

Abstract
This research explores the development of a novel trespasser detection architecture and feedback system for school surveillance purposes, with the aim of reducing the laboriousness of this currently manual and inefficient process. This was achieved by feeding images captured by an Autonomous Land Vehicle to a pose estimation network to detect the presence of trespassers, then feeding the resulting output to a pose classification network to detect the action of the predicted person, with the aim of providing information as to their intentions. This information is sent to security personnel over the messaging platform, Telegram, allowing them to monitor a scene remotely. Overall, results suggest that this method was highly promising in detecting a human presence, with low latencies of 1.99s on average.

1. Introduction

1.1 Background and Purpose of Research

School surveillance is a highly laborious process, a problem amplified by the large size of modern campuses. Given the large surveillance area of modern campuses, it is physically exhausting for the guards to cover the school grounds manually: a method both highly inefficient and ineffective.

Although most campuses have installed closed circuit television cameras (CCTVs) to reduce the inefficiencies in school surveillance, the effectiveness of these cameras are limited given that they are placed at fixed locations. Moreover, naively increasing the number of CCTV cameras scales poorly given the number of additional manpower needed to monitor the increased number of CCTV feeds, resulting in this approach to be prohibitively expensive at scale.

A solution to that would be to employ autonomous surveillance vehicles, which have become popular with the inception of self-driving. Unlike the traditional approach of using CCTVs, ALVs can move autonomously and cover a larger surveillance area. However, most robotics platforms cost upwards of thousands, which adds significant cost considerations into implementing these technologies in schools. To that end, we propose using low-cost Autonomous Land Vehicles (ALVs) for school surveillance. The proposed ALV can detect persons and send notifications when persons are detected, and reduce idle time used to monitor camera feeds.

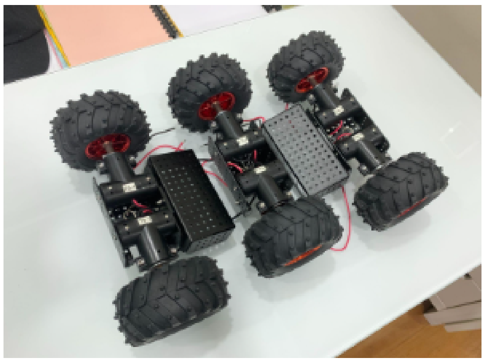


Fig. 1: *6-wheeled vehicle*

Our report is broken down into the following sections: Section 2 details the materials and methods behind the hardware and software components of our project, while Section 3 delves into a discussion of the results obtained when this system was implemented.

At a high-level, this school security architecture integrates an existing autonomous 6-wheeled vehicle developed by a senior (Fig. 1), which can perform a basic line following tasks, with PID capabilities.

1.2 Literature Review

To the best of our knowledge, the few smart surveillance systems in the market make use of object detection networks to identify the presence of anomalies in a vicinity (Goldmeier, 2022). While suitable for surveillance, it fails to provide specific information as to the intentions of the person(s) identified, if any. In addition, the accuracy of such a model falters in certain conditions and environments, such as dimly-lit ones, or when image quality is low, unless specifically trained with a large dataset (Kumar et al, 2022). To better counter this issue, this project sought to test the efficacy of using a pose estimation network instead. Pose estimation is a computer vision technique that predicts a pose of a person, allowing security to be able to have a sensing of what the person may be doing and their intentions. Furthermore, since the pose detection network extracts pose estimates as inputs to train pose classification network eventually, it is invariant to whether the images are coloured or grayscale, making it a greater suit for this application. As pose estimation only classifies poses of humans, it would serve an augmented purpose of human detection, aiding with intrusion detection.

2. Materials and Methods

The integrated system takes on the role of a patrolling personnel and notifies security guards with its feedback system via the telegram bot. As such, the school security is able to monitor multiple locations around the school remotely and simultaneously. Additionally, instead of requiring security personnel to idly monitor our robots camera feed, our architecture actively uses pose estimation to detect the presence of intruders, allowing security guards to discern the best possible course of action to take, saving both time and energy. We show our surveillance architecture in Figure 2. When activated by security personnel, our ALV platform patrols the demarcated areas in the school compound via line-tracing, and detects potential trespassers along its patrol route (Appendix A). When trespassers are detected, the trespasser's activity i.e. walking or jogging is detected using the process outlined in Figure 3, and the security personnel is notified over telegram for follow-up actions.

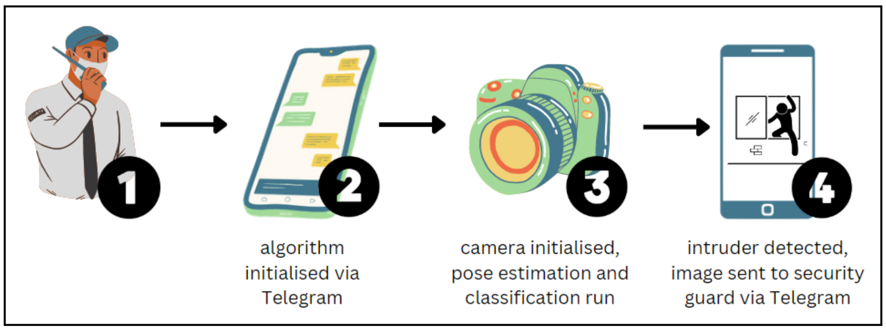
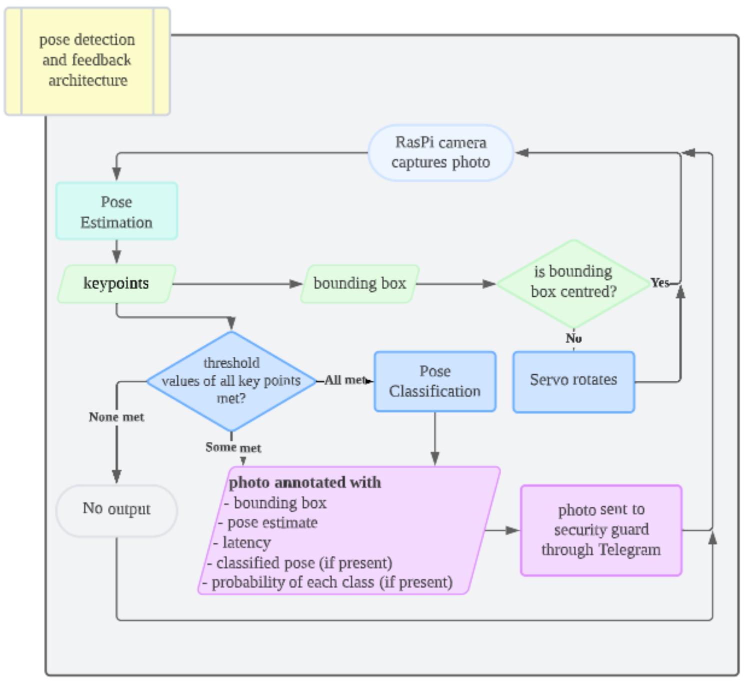

1
2
algorithm initialised via Telegram
3
camera initialised, pose estimation and classification run
4
intruder detected, image sent to security guard via Telegram

Fig. 2: *Overview of surveillance system*



pose detection and feedback architecture
RasPi camera captures photo
Pose Estimation
keypoints
bounding box
is bounding box centred?
Yes
No
Servo rotates
threshold values of all key points met?
All met
None met
Some met
Pose Classification
No output
photo annotated with
- bounding box
- pose estimate
- latency
- classified pose (if present)
- probability of each class (if present)
photo sent to security guard through Telegram

Fig 3: *Pose Detection and feedback architecture*

2.1 Hardware

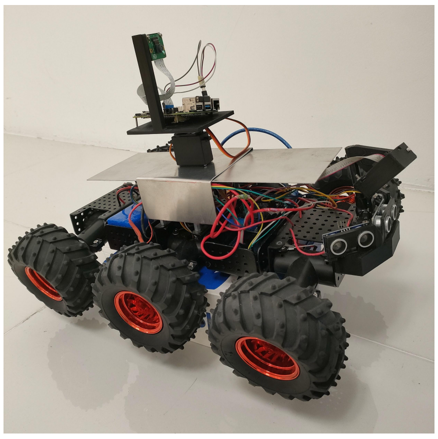


Fig. 4: *Hardware overall setup with ALV*

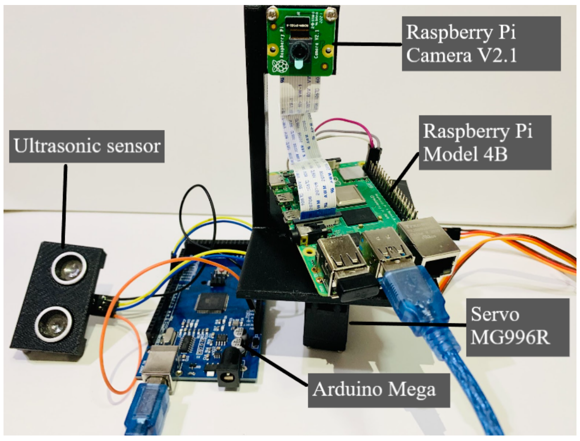


Fig. 5: *Hardware setup for pose detection and feedback capabilities*

We detail our ALV's hardware system in Fig. 4-5. We use a RaspberryPi 4 Model B with a 32-bits operating system for its fast inference speed and low power consumption through its ARM microprocessor. This makes it suitable for the processing of machine learning algorithms, as in our use case (Appendix B). The 32-bits operating system was used instead of the 64-bits due to the lack of open-source libraries available for the latter. We use a 32-bit operating system (OS) over the more recent 64-bit OS because the former 32-bit OS has better support. A Raspberry Pi Camera V2.1 is connected to the Raspberry Pi to capture images and a MG996R Servo is used to help the camera track potential trespassers. The Raspberry Pi communicates with an Arduino Mega 2560 microcontroller, which interfaces with the ALV's remaining sensors and actuators e.g., ultrasonic sensor and motors.

We integrate the Raspberry Pi Camera and Servo using a 3D printed platform (Appendix C).

2.2 Software

To detect trespassers, our ALV uses pose estimation to obtain the pose of the trespasser. These pose estimates are used to generate a bounding box around the trespasser and to determine the activity of the person using a pose classification network.

2.2.1 2D Pose Estimation

2D Pose Estimation is the task of localising the key joints of a person(s) e.g. elbows, knees, in an image. We use MoveNet to detect the landmark coordinates of 17 key joints of the human body to perform pose estimation (Fig. 6). MoveNet is a bottom-up model that relies on MobileNet V2 as a feature extractor (Votel & Na, 2021). Although it has a lower accuracy than other popular networks such as OpenPose or PoseNet, it offers real-time performance (Jo & Kim, 2022), with lower latencies than the former neural networks (Appendix D).

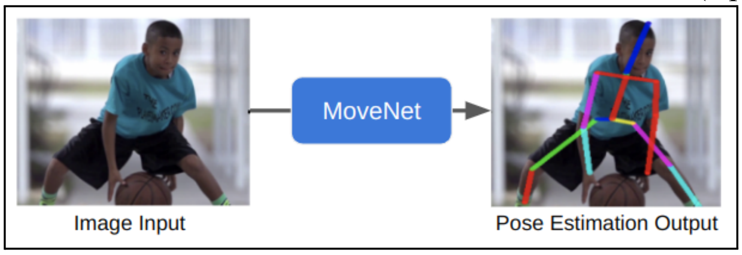


Fig. 6: *Overview of MoveNet Pose Estimation*

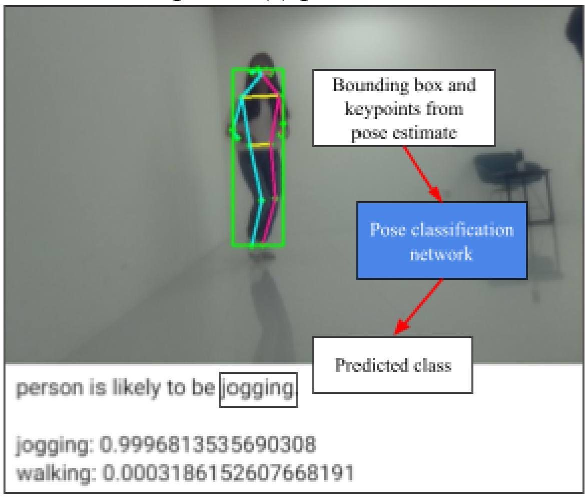
2.2.2 Person Detection

The task of person detection involves localising the positions of person(s) in an image. From our estimated poses in Section 2.2.1, we generate the bounding box around person(s) by letting the highest and lowest *x* and *y* key joint coordinates denote the 4 corners of the box.

These detections are fed into a simple servo rotation algorithm which rotates the camera and centres the largest detected person in the camera frame. The code for this can be found in Appendix E.

2.2.3 Pose Classification.

Pose classification is the task of classifying the person(s)' pose in an image. We train a simple Multi-layer Perceptron network on the pose estimation outputs from MoveNet (Fig 7).Our pose classification network classifies person(s) poses into the classes: walking and jogging.


Bounding box and keypoints from pose estimate
Pose classification network
Predicted class

person is likely to be jogging

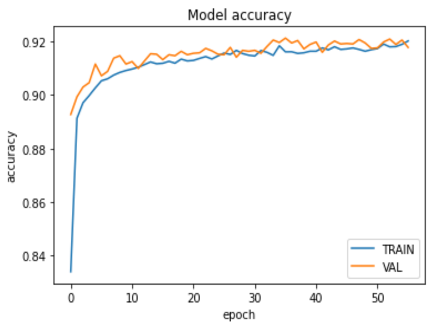
jogging: 0.9996813535690308
walking: 0.0003186152607668191

Fig. 7: *Keypoints and resulting bounding box from a Pose Estimate*

We use the publicly available KTH Action Dataset, consisting of mono-channel videos divided into people walking and jogging (Laptev et al, 2005), to train our pose classification network. We extracted each frame of the video into images and used the video labels i.e. walking/jogging to label corresponding images. The entire dataset consists of approximately 46000 video images (Fig. 8). We split the dataset into a 80/20 train-test split. The exact number of images in each split can be found in Appendix F.

Fig. 8: *Sample images from KTH Action Dataset*

Given that our pose classification dataset from KTH only contains mono-channel images while our test images during deployment are RGB images, there is a domain gap between these two image representations. To enable our pose classification network to be invariant to either mono-channel or RGB images, we extract domain-invariant pose estimations using MoveNet from each image, and use these pose estimations as the input into our classifier model.



A total of 56 epochs were trained when the accuracy of the validation data set no longer increased significantly after 20 epochs. The model accuracy achieved when ran on the validation set was **92.5%**. To ensure that we did not overfit, we observed the model's validation accuracy trend.

Fig. 9: Model accuracy for validation and training dataset

3. Results and Discussion

3. 1 2D Pose Detection

To evaluate the effectiveness of the Pose Estimation and Pose Classification networks, testing was conducted on 2 different datasets: a subset of test images from the KTH action database (Appendix G), as well as a custom set of 233 images taken in a real-world setting to simulate the actual location in which the robot would be deployed. In this section, the results of the algorithm when deployed on each dataset will be analysed in detail.

3. 1. 1 Pose Estimation

On both datasets, key points in images were generally identified accurately. The algorithm was able to detect the presence of people in a vicinity, even when they were some distance away, or positioned at the edge of the camera frame.

Some of the challenges of pose estimation include the accuracy of variability of human visual appearance in images, partial occlusions due to self articulation and layering of objects in the scene or high dimensionality of the pose (Sigal. L, 2016). However, when we analysed the pose estimate of the photos under a few different cases under different conditions (i.e the person possessing objects, person not entirely in the frame, noises in vicinity), we noted that the pose estimates were still largely accurate under these conditions (Fig. 10-12).

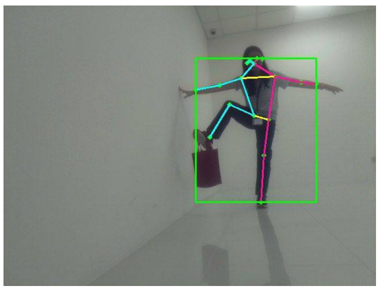


Fig. 10: Person possessing an object

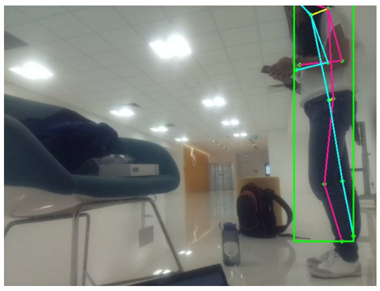


Fig. 11: Person partially occluded

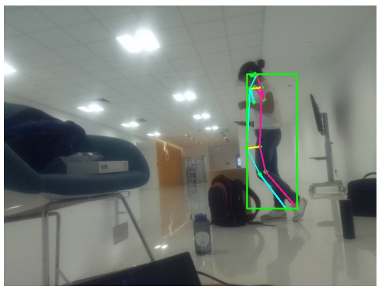


Fig. 12: Noise in the vicinity

Further samples of successful estimates can be found in Appendix H. We found that the pose estimation algorithm generalised well to a real life setting, being able to detect and identify key points from a variety of poses, and in cluttered and clear backgrounds. This suggests that the algorithm is largely unaffected by noise, which is key to the deployment of our architecture in a real-world setting where background objects will be more abundant. This obliviousness to noise indicates the reliability of the architecture in aiding with surveillance.

Notably, the fact that MoveNet is able to perform pose estimation very accurately on extreme actions is one of the reasons for choosing this network. As it was trained on COCO and an internal Google dataset (Active) that contains a high number of challenging fitness poses while also experiencing significant motion blur effect, it performs pose estimation much more accurately than identical architectures that solely trained with COCO dataset (Votel R., Li N., 2021). As the resolution of photos captured by the Raspberry Pi camera would not be high and intruders may be involved in extreme actions such as running or violent actions, we simulated a few of such actions resulting in less defined photos and found that the network was generally able to perform pose estimation accurately in such cases.

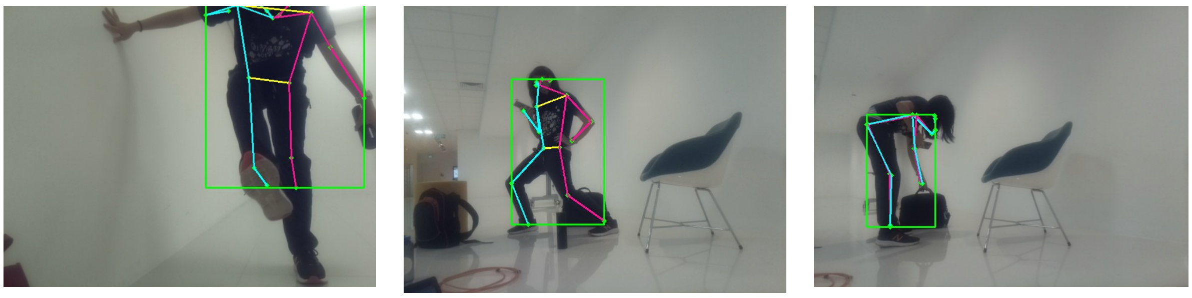


Fig. 13: *Extreme actions*

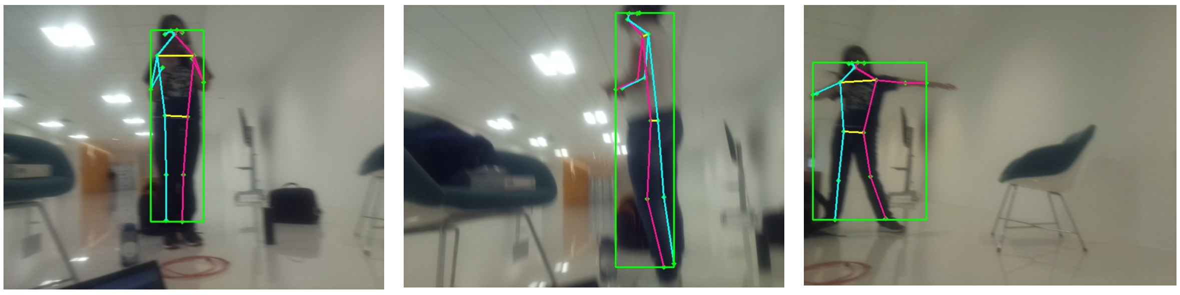


Fig. 14: *Blurry images yet accurate prediction*

That being said, there are occasions when key points are misidentified and placed on inanimate objects, such as empty spaces, or on chairs. As MoveNet is a bottom-up model (Jo B.J, Kim S.K, 2022), with its perks including the absence of biases in data, there are also its downsides of occasionally misidentifying or inaccurately annotating key points while a person is visibly and sufficiently in frame. This is likely because the bottom-up approach detects all parts of each person within an image and then associates parts that belong to each individual instead of the top-down approach that first bounds up a person by incorporating a person detector first, estimating the location of body parts, and finally calculating a pose for each person. The failure cases are presented below in Fig. 15, 16, 17.

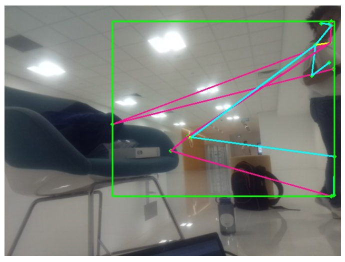


Fig. 15: *overlap of keypoints with inanimate objects when the person is partially out of frame*

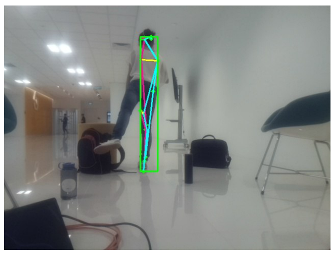


Fig. 16: *key point of the person inaccurately annotated when the person's leg overlaps with an inanimate object*

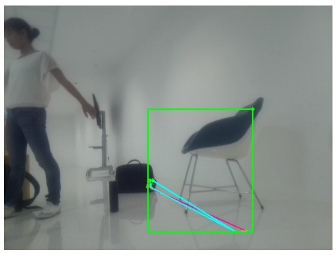


Fig. 17: *Person is completely missed*

3.3 Pose Classification

When performed on the test dataset, the pose classification model yielded a mean accuracy of **92.5%**, with the highest accuracy in predicting instances of walking (Fig. 18). This is likely influenced by the similarity of conditions and poses performed by the same person between the training and test set.

Conversely, accuracy dropped when the model was deployed on our custom dataset, with an accuracy of **42.5%** (Fig. 19). This could be due to the fact that the majority of images in the training set used were ones in which the subject matter was positioned sideways to the camera, where the difference in arm movements when walking or jogging were more stark. An analysis suggests that the pose classification relies heavily on the subject matter's arm position: akin to how people swing their arm while walking and keeping their hands bent and closer to their bodies when running, the algorithm tended to classify images where the subject matter had their arms apart as 'walking', while images where the subject matter held their arms closer to their body were classified as 'jogging' (Fig. 20). However, when applied to images where the subject matter faced the camera, this led to inaccurate classifications due to an overreliance on this detail.

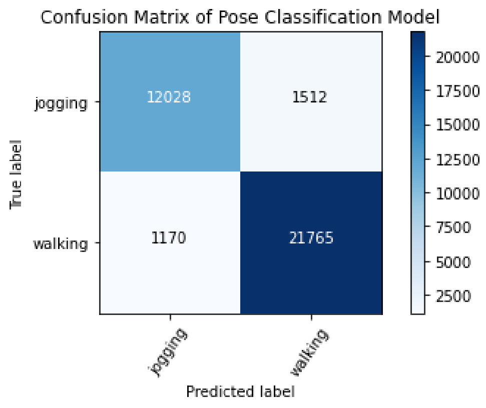
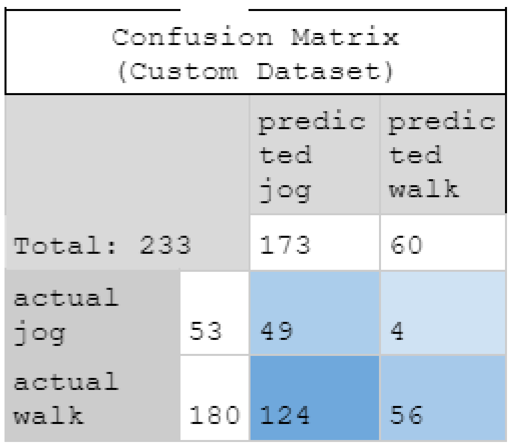


Fig.18: *Confusion matrix of pose classification model*



Confusion Matrix (Custom Dataset)			
		predicted jog	predicted walk
Total: 233		173	60
actual jog	53	49	4
actual walk	180	124	56

Fig.19: *Confusion matrix (custom dataset)*

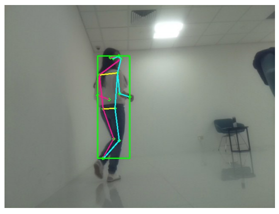
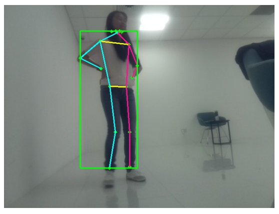
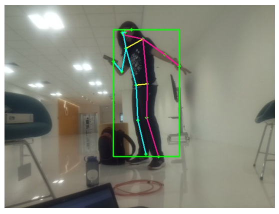
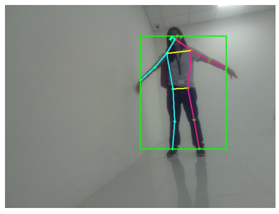
			
True Positive (jogging)	False Positive (jogging)	True Positive (walking)	False Positive (walking)

Fig. 20: *Sample of true/false positive cases for jogging and walking*

Indeed, when we only considered images where the test subject was positioned sideways to the camera, accuracy improved, to **58.2%** (Fig. 21).

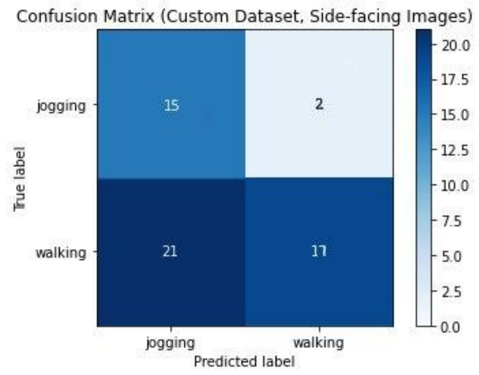


Fig. 21: *Confusion Matrix (side-facing images in custom dataset)*

To improve these results, further training could be conducted on a larger dataset depicting people jogging and walking from more varied camera angles, to improve the accuracy of the pose classification algorithm when performing on new data. The application of transfer learning to augment images in the training set could also be considered, allowing for a computationally efficient way of improving the model's accuracy (Zhuang et al, 2009).

3.2 Latency

To test the latency of our architecture, we ran the algorithm 244 times on our custom dataset, collecting information on the latency of each successful run – i.e. one where all key points were detected and the poses were classified. This gave us a minimum of **0.98** seconds and a mean of **1.99** seconds. Although significantly higher latencies of 3-4 seconds were experienced during the initialisation of the algorithm via a telegram message, the majority of classifications had lower latencies, allowing for security personnel to receive close-to real-time data from a scene.

3. 3 Servo rotation

The servo centres the bounding box identified, thus preventing keypoints from being occluded and allowing for a more accurate classification (Fig. 22-23).

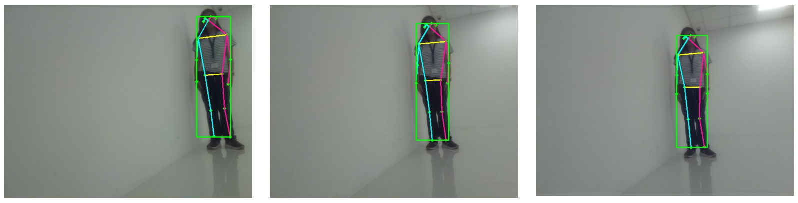


Fig. 22: *Servo rotation to the left*

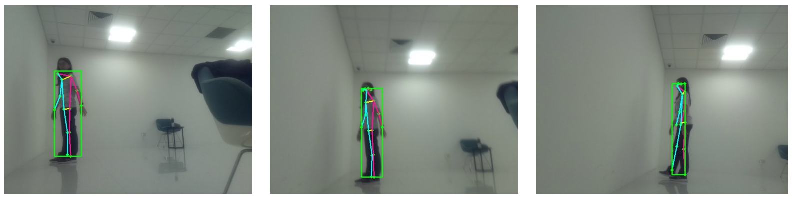
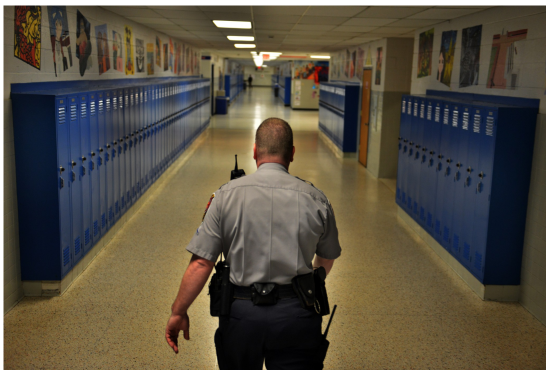


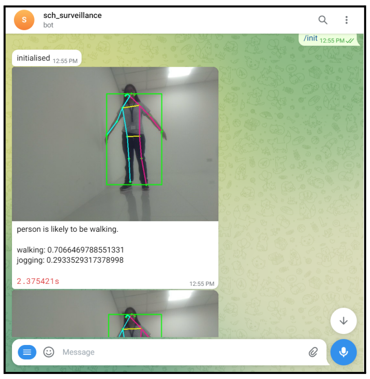
Fig. 23: *Servo rotation to right*

However, lags experienced occasionally cause the servo to over or undercompensate when rotating the camera to centre the bounding box.

Overall efficacy

While there exists room to improve the accuracy of the models, this architecture functions as a dependable surveillance system, largely able in detecting the presence of people in the vicinity. A comparison of the surveillance process reveals the efficiency of our architecture in automating the process, when applied on a telegram bot. Instead of traversing the school campus to seek out intruders, this highly efficient system detects and sends images of potential intruders to the security guards directly at relatively low latencies, allowing them to monitor areas of the school remotely, saving both time and energy (Fig. 24).





sch_surveillance
bot

/init 12:55 PM

initialised 12:55 PM

person is likely to be walking.

walking: 0.7066469788551331
jogging: 0.2933529317378998

2.375421s 12:55 PM

Before: *Security patrols school frequently overnight*

After: *Security is able to remotely monitor the school campus*

Fig. 24: *Surveillance process before/after architecture*

4. Conclusion and Future Works

In summary, this project explores the development of a low-cost system for surveillance purposes that integrates existing technologies to create a highly accurate trespasser detection system with a Telegram-based notification module. Our approach reduces the laboriousness of school surveillance and offers a more efficient means of detecting trespassers when compared to existing methods where surveillance videos have to be filtered through manually to identify the presence of intruders. The added comprehensiveness that comes with using pose estimation and classification techniques further enhance the system, setting it apart.

We offer the following suggestions for exploration to expand upon our work:

Firstly, with regards to improving efficiency and accuracy of the servo's tracking algorithm, a new network could be implemented. Currently, the servo motor attached to the camera's motion currently makes use of a 'trial and error' method when rotating the camera to focus on people in the vicinity. Efficiency could be improved by calculating the exact distance and angle of a person with respect to the camera, thereafter rotating the servo accordingly. This could be achieved using a Monocular Depth Estimation network, which aims to infer 3-dimensional space geometry, such as the depth value of each pixel in a single 2-dimensional still image (Basu, 2021). This would inevitably allow for more accurate footage, as well as more accurate information on the movements of persons detected, to be obtained, but such a network would require large amounts of ground truth depth data to be collected and hence was unfeasible to implement at the time of our project.

Thirdly, different architectures (such as the PoseNet or OpenPose models) could potentially be used to increase the accuracy of detections made: while this could mean a compromise on latency, further experimentation would be needed to truly ascertain the extent of this impact to our use case.

Fourthly, the current algorithm assumes only one person in the camera's range of vision. While this is likely a reasonable assumption given our use case of school surveillance during the night, where passersby nor crowds should be present, there may be occasional cases where more than one person enters the frame. To account for this, alterations would have to be made to the pose detection network utilised.

Lastly, a deep sort algorithm which predicts the trajectory of a person by utilising Kalman filters, Intersection over Union (IoU) or other methods could be implemented, this would allow for more information to be given to security with regards to where the intruder might be headed to. However, the concern about the limited processing power of the raspberry pi to process so many networks simultaneously surfaces.

Nevertheless, this project has led to the development of a viable surveillance and feedback system by means of pose estimation and classification networks, creating a foundation upon which future work can be built.

5. Acknowledgments
We would like to thank our mentor, Mr Kennard Ng, for his invaluable guidance throughout the course of this project. We would also like to extend our gratitude to Ms Joyce Yao and Dr Kelvin Tan for their administrative support and for providing us with this enriching opportunity.

6. References

Basu, V. (2021, August 30). Keras Documentation: Monocular depth estimation. Keras. Retrieved December 27, 2022, from https://keras.io/examples/vision/depth_estimation/

Goldmeier, L. (2022, March 29). Object detection and identification in video analytics. BriefCam. Retrieved December 31, 2022, from https://www.briefcam.com/resources/blog/object-detection-and-identification-in-video-analytics/

Google image result for https://smartsecuritypros.com/wp-content/uploads/2017/03/school-security.jpg. (n.d.). Retrieved January 1, 2023, from https://images.app.goo.gl/2XKJPbKqwViwnQ5t5

Kumar, A., Talati, B., Rajput, M., & Trivedi, H. (2022). (tech.). A Novel Approach to Low Light Object Detection Using Exclusively Dark Images. Association for Computing Machinery. Retrieved December 31, 2022, from https://doi.org/10.1145/3533050.3533064.

Laptev, I. & Caputo, B. (2005, January 18). Recognition of human actions. Retrieved December 27, 2022, from https://www.csc.kth.se/cvap/actions/

Maslov, M. (2021, June 29). Security Robots Outdoor Patrolling. SMP Robotics - Autonomous mobile robot. Retrieved December 30, 2022, from https://smprobotics.com/security_robot/

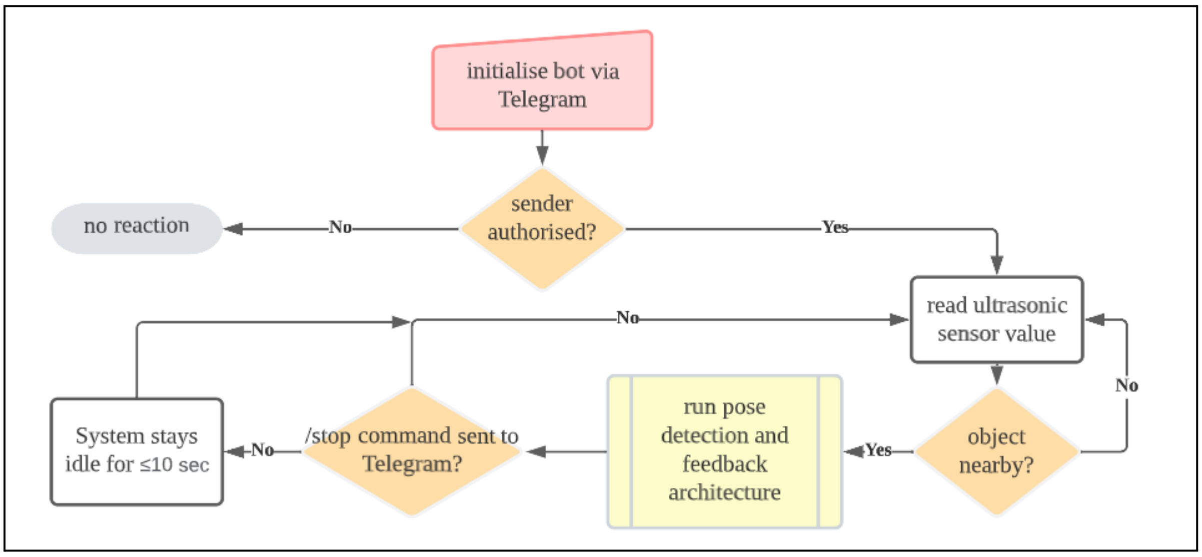
MoveNet: Ultra fast and accurate pose detection model. TensorFlow. (2022). Retrieved December 31, 2022, from https://www.tensorflow.org/hub/tutorials/movenet

Votel, R., Na, L. (2021, May 17). Next-generation pose detection with movenet and tensorflow.js. The TensorFlow Blog. Retrieved December 31, 2022, from https://blog.tensorflow.org/2021/05/next-generation-pose-detection-with-movenet-and-tensorflowjs.html

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Hui, X., Qing, H. (2009, December 17). A Comprehensive Survey on Transfer Learning . arxiv. Retrieved December 30, 2022, from https://arxiv.org/pdf/1911.02685v2.pdf

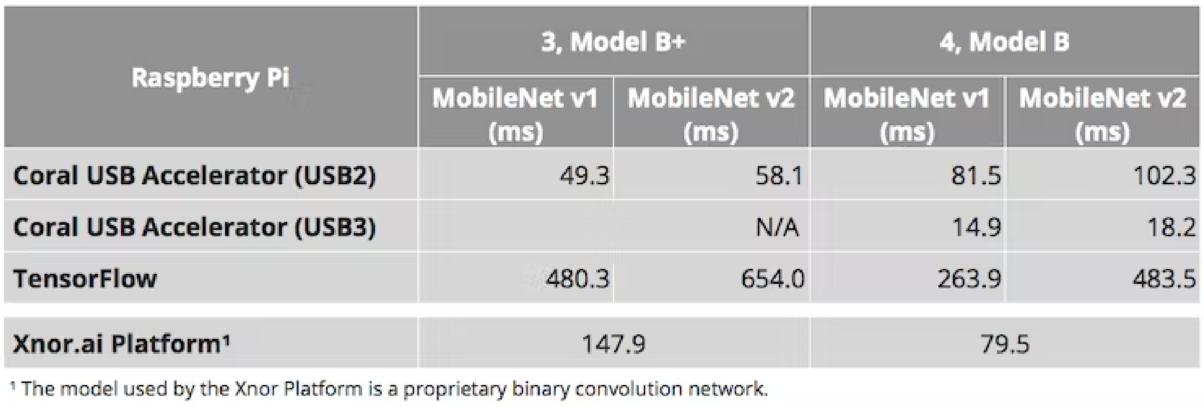
Appendices

Appendix A


initialise bot via Telegram
sender authorised?
No
no reaction
Yes
read ultrasonic sensor value
No
object nearby?
No
Yes
run pose detection and feedback architecture
/stop command sent to Telegram?
No
System stays idle for ≤10 sec

Architectural diagram of the School Surveillance System

Appendix B

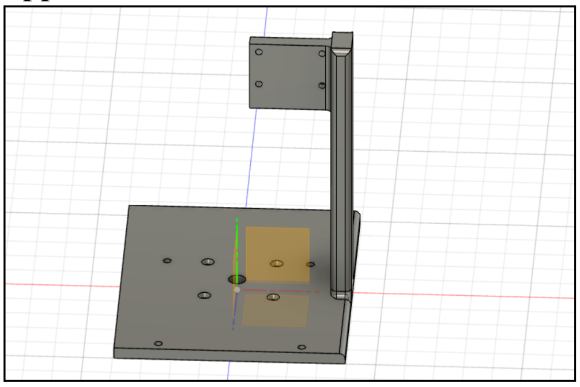


Raspberry Pi	3, Model B+		4, Model B	
	MobileNet v1 (ms)	MobileNet v2 (ms)	MobileNet v1 (ms)	MobileNet v2 (ms)
Coral USB Accelerator (USB2)	49.3	58.1	81.5	102.3
Coral USB Accelerator (USB3)		N/A	14.9	18.2
TensorFlow	480.3	654.0	263.9	483.5
Xnor.ai Platform[1]	147.9		79.5	

[1] The model used by the Xnor Platform is a proprietary binary convolution network.

Comparison between Raspberry Pi Model 3 and 4 (Allan A., 2018)

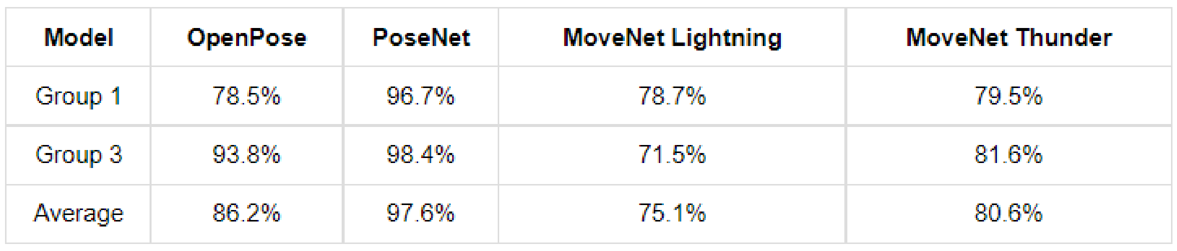
Appendix C



3D printed platform for Raspberry Pi, Raspberry Pi camera and servo

Appendix D

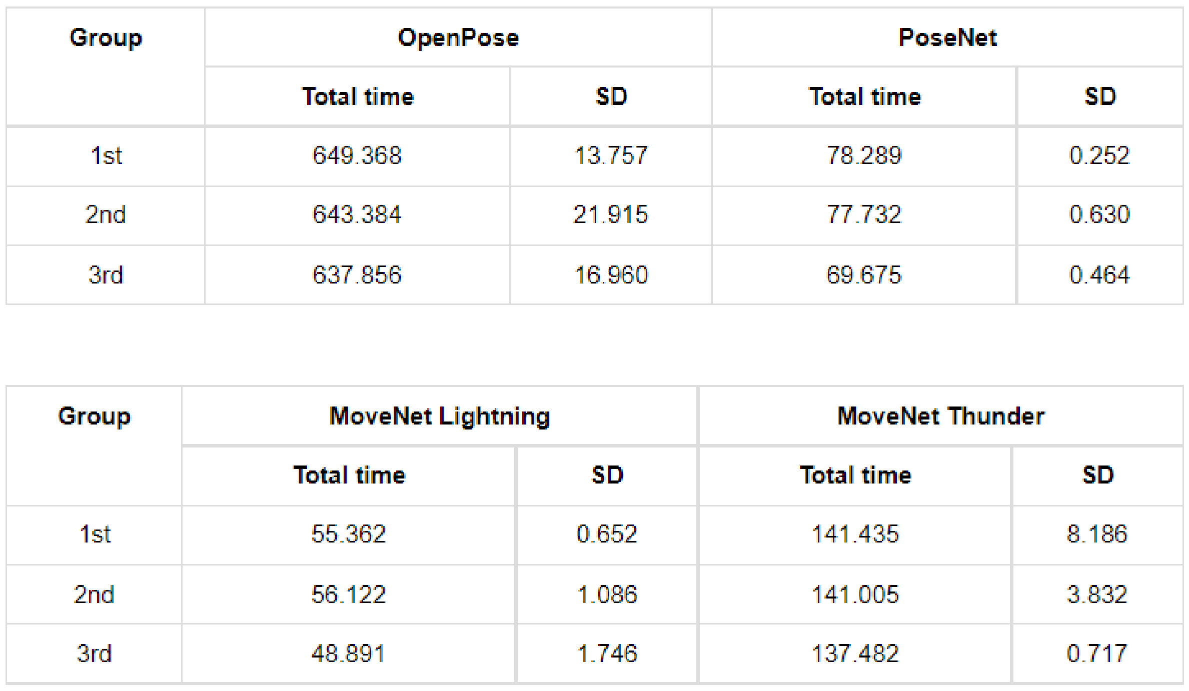
Table 4. Pose estimation accuracies on images within the first and third groups



Model	OpenPose	PoseNet	MoveNet Lightning	MoveNet Thunder
Group 1	78.5%	96.7%	78.7%	79.5%
Group 3	93.8%	98.4%	71.5%	81.6%
Average	86.2%	97.6%	75.1%	80.6%

Comparison of accuracy between OpenPose, PoseNet, MoveNet Lightning & MoveNet Thunder (Jo & Kim, 2022)

Table 2. Total time (unit: s) to estimate 1,000 images within a group and its standard deviation



Group	OpenPose		PoseNet		MoveNet Lightning		MoveNet Thunder	
	Total time	SD	Total time	SD	Total time	SD	Total time	SD
1st	649.368	13.757	78.289	0.252	55.362	0.652	141.435	8.186
2nd	643.384	21.915	77.732	0.630	56.122	1.086	141.005	3.832
3rd	637.856	16.960	69.675	0.464	48.891	1.746	137.482	0.717

Comparison of latency between OpenPose, PoseNet, MoveNet Lightning & MoveNet Thunder (Jo & Kim, 2022)

Appendix E

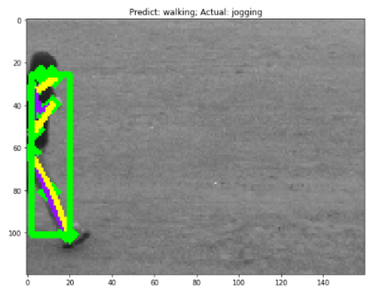
```python
"""servo movement"""
# find x-position of person and compare with camera center
person_x = (
    person.bounding_box.end_point.x - person.bounding_box.start_point.x
)
logging.debug(f"person x-position: {person_x}")

# if person is to the left of the camera, turn servo to the right
if person_x < CAM_X - ANGLE_THRESHOLD:
    ANGLE += 10
    if ANGLE > 180:
        ANGLE = 180
        await context.bot.send_message(
            CHAT_ID, "angle > 180, automatically set to 180 instead"
        )

# if person is to the right of the camera, turn servo to the left
elif person_x > CAM_X + ANGLE_THRESHOLD:
    ANGLE -= 10
    if ANGLE < 0:
        ANGLE = 0
        await context.bot.send_message(
            CHAT_ID, "angle < 0, automatically set to 0 instead"
        )

# set servo angle
await set_angle(ANGLE)
```

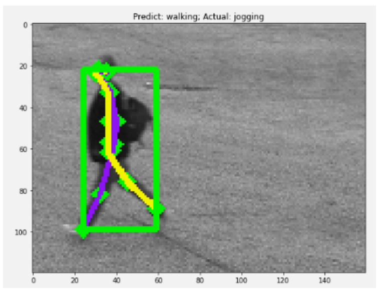
Appendix F

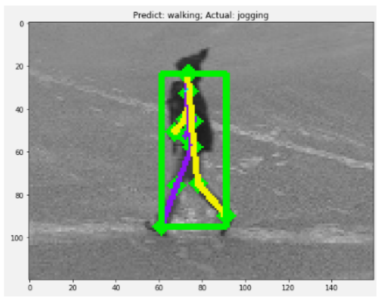
	Train *(80%)*	Test *(20%)*	Total
Jogging	*13538*	*3406*	*16944*
Walking	*23291*	*5762*	*29053*

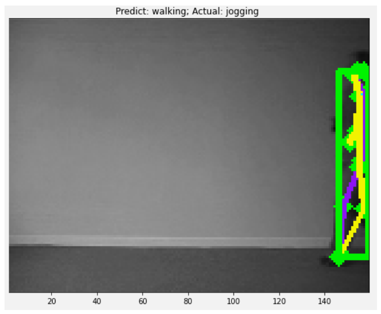
Breakdown of number of images for training and test sets

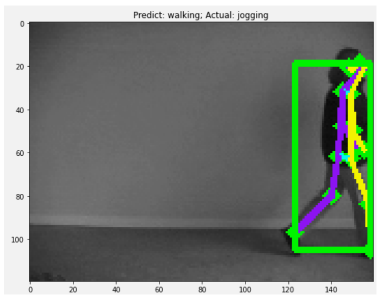
Appendix G

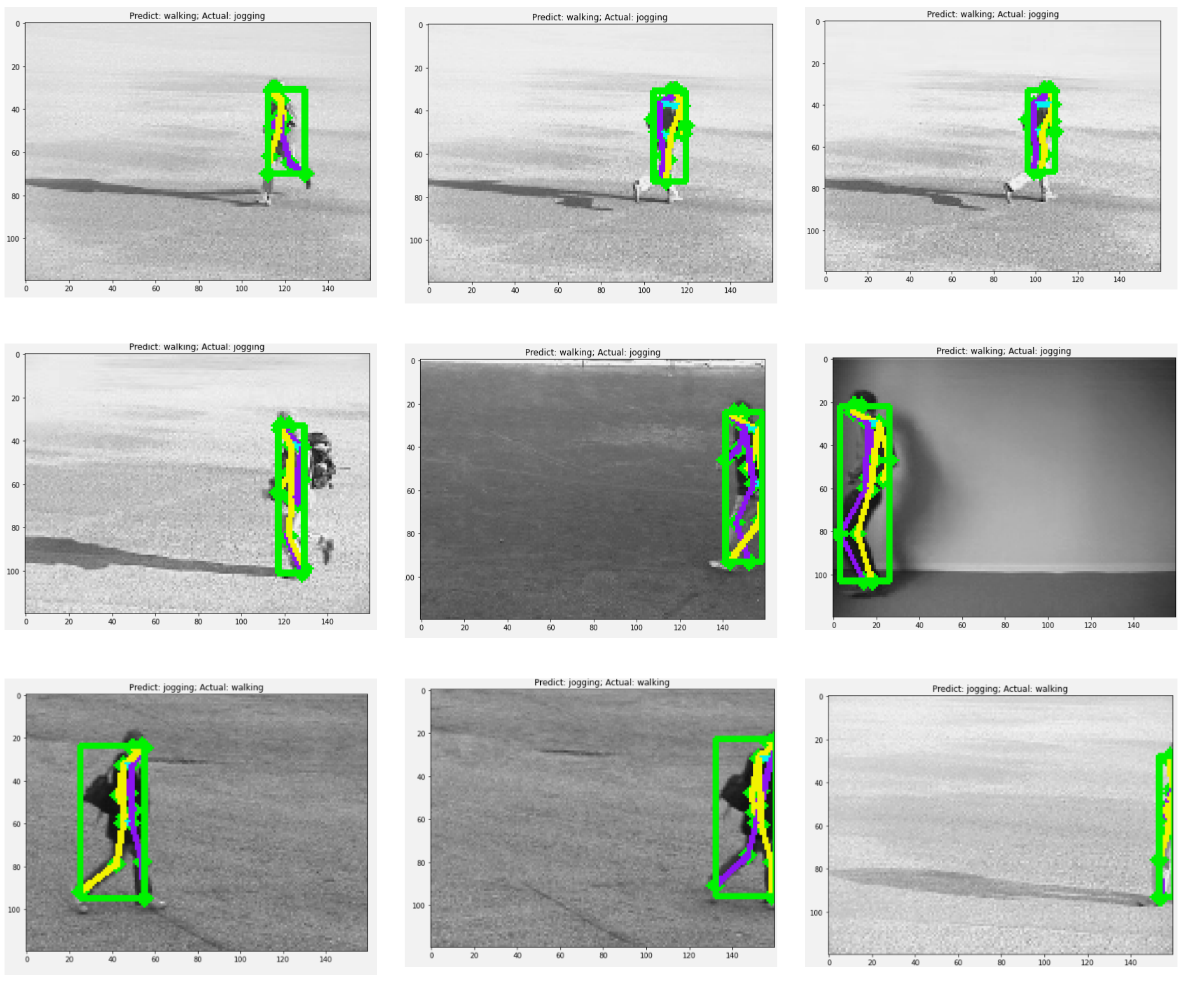






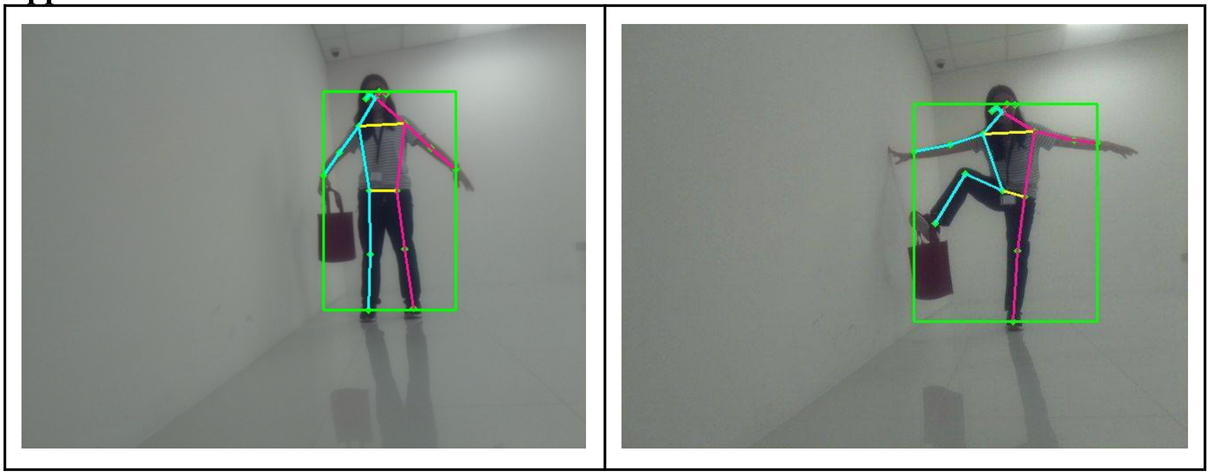




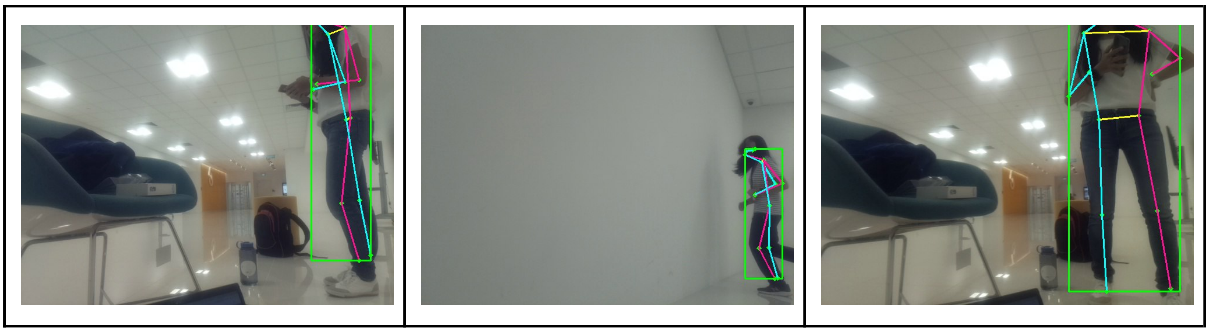

Predict: walking; Actual: jogging
Predict: walking; Actual: jogging
Predict: walking; Actual: jogging
Predict: walking; Actual: jogging
Predict: walking; Actual: jogging
Predict: walking; Actual: jogging
Predict: jogging; Actual: walking
Predict: jogging; Actual: walking
Predict: jogging; Actual: walking

Test set falsely classified images

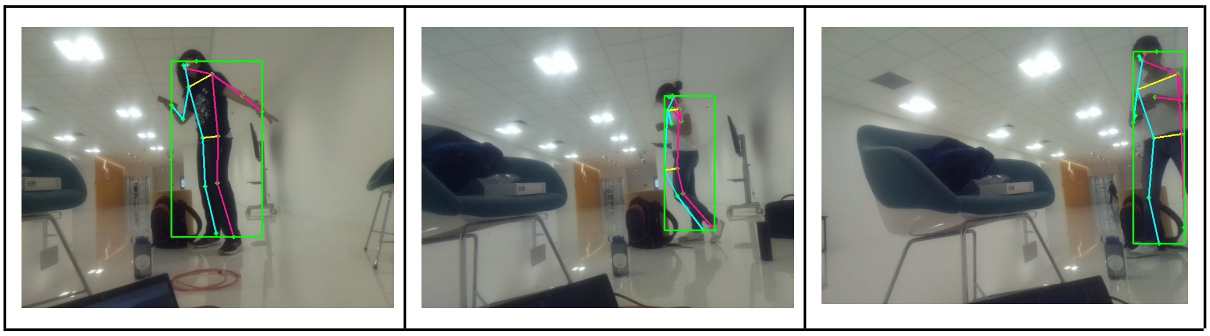
Appendix H



Case 1: *Person possesses objects*



Case 2: *Person is not completely shown in frame*



Case 3: *Noise within the vicinity*