

An Investigation into Real-time Tracking and Reidentifying of 3D Objects in a Multi-Spectrum, Multi-Camera System

BS Steyn

(Department of Computer Science)

Stellenbosch University

21740178@sun.ac.za

Abstract—Object detection is one of the most relevant fields in the current landscape of computer science. This has led to many techniques that have been developed to solve the challenges involved in this field. The challenge this paper focuses on is applying techniques to a multi-spectrum image landscape. This paper compares a Siamese Neural Network with standard image embedding in both the normal and infrared light spectrums to determine if these techniques are viable for multi-spectrum use, as well as determining if standard detection libraries like YOLO are applicable in a multi-spectrum landscape. The results demonstrate that these techniques are applicable, however there are challenges with the application in the multi-spectrum landscape.

I. INTRODUCTION

In the era of smart surveillance, autonomous vehicles and intelligent urban systems, reliably tracking and reidentifying objects across video streams has become a cornerstone of modern computer vision applications. Real-time object tracking and reidentification (Re-ID) [3] aim to monitor and consistently recognize objects, such as pedestrians, vehicles, or luggage - across frames - even as they move through different viewpoints or reappear after temporary occlusion. This task is challenging due to real-world complexities, such as changes in illumination, occlusions, variations in appearance due to different angles, and movement across multiple non-overlapping cameras.

Real-time object [21] tracking typically involves locating an object in sequential frames, and maintaining continuous identification as it moves. Reidentification [3], on the other hand, requires matching instances of the same object across frames, cameras, or even diverse environments, enabling consistent recognition when it reappears after disappearing from the field of view. The convergence of these two tasks into a single framework is essential for applications where both continuity and accuracy in object identity are paramount.

Recent advancements in deep learning, especially convolutional neural networks (CNNs) [17], vision transformers (ViTs) [9] and Siamese networks [14], have significantly enhanced feature extraction capabilities for reidentification. These models excel at capturing subtle spatial and appearance-based features that remain consistent across varying perspectives. Additionally, metric learning methods, such as triplet loss and contrastive loss, have further strengthened Re-ID models [3] by allowing them to effectively distinguish between instances of similar objects and better align feature representations for identical objects.

However, achieving real-time performance in such complex scenarios requires balancing accuracy with computational efficiency [8]. Real-time implementations must account for system constraints, prioritizing rapid response times while maintaining robust reidentification accuracy. Hybrid approaches - combining traditional tracking methods such as Kalman filtering [12] with deep-learning-based feature extraction - have shown promising results, providing a feasible balance between speed and reliability. For example, systems like DeepSORT [22] integrate deep feature embedding with traditional tracking algorithms, achieving real-time, multi-object tracking performance by incorporating spatial and temporal features.

The remainder of this paper is organised as follows: Section II provides an overview of the various algorithms and reidentification techniques investigated in this paper. Section III provides an outline for our investigation of the different algorithms and techniques. Section IV presents the implementation details for this paper, the dataset is also discussed. Section V presents the main findings of this paper which is subsequently summarised and concluded in Section VI.

II. BACKGROUND

This section provides an overview of the main algorithms used in this paper.

A. Kalman Filter

In 3D tracking, the Kalman filter [12] plays a critical role in accurately estimating the position and velocity of objects in real time, especially when measurements are noisy or incomplete. This capability is foundational in applications such as augmented reality (AR), virtual reality (VR), motion capture, and autonomous systems, where accurate tracking in a three-dimensional space is essential for performance and user experience. There are some general challenges with 3D tracking they are listed below:

- **Measurement Noise:** Sensors used in 3D tracking, such as LiDAR, radar, and computer vision systems, are prone to noise due to environmental factors (e.g., lighting, weather conditions) and inherent limitations in sensor resolution.
- **Non-Uniform Motion:** Objects in 3D space often move non-linearly or at variable speeds, which can complicate the estimation process.
- **Occlusions and Missing Data:** In complex environments, objects may become occluded or partially observed, leading to gaps in data that must be managed for continuous tracking.

The Kalman filter addresses these challenges by providing a probabilistic approach that accounts for both process dynamics and measurement uncertainty. The Kalman filter is used to track the state of an object—its position, velocity, and sometimes acceleration—in 3D space. The state vector typically includes the x, y, and z coordinates, along with their respective velocities. The filter operates by recursively predicting the object's next state based on its previous position and velocity and then updating this prediction as new measurements arrive.

- **Prediction:** Using a model that incorporates previous estimates of position and velocity, the filter predicts the object's next state in 3D space. This step relies on known system dynamics, which could be linear or non-linear, depending on the object's movement pattern.
- **Update:** When new 3D positional measurements are available from sensors (e.g., stereo cameras or depth sensors), the filter updates its prediction by balancing between the forecasted position and the observed measurement. The weighting is influenced by the confidence in each source, as captured by covariance matrices that quantify measurement noise

and process uncertainties. The main advantages of the Kalman filter are:

- **Smoothing Out Noisy Measurements:** The Kalman filter mitigates the effects of measurement noise, resulting in smoother tracking trajectories.
- **Handling Occlusions:** By predicting the object's position, the filter can continue to estimate the location even if the object is temporarily occluded.
- **Supporting Real-Time Processing:** The recursive update process is computationally efficient, which is essential in applications requiring immediate responses, such as autonomous driving and AR/VR.

In summary, the Kalman filter and its variants offer a robust, efficient solution for 3D tracking, providing reliable state estimation even in noisy, unpredictable environments. Its use in this domain is critical for enhancing the stability and precision of tracking systems in applications where real-time accuracy is paramount.

B. Hungarian Algorithm

The Hungarian algorithm [15], also known as the Kuhn-Munkres algorithm, is a combinatorial optimization algorithm that solves assignment problems efficiently by finding the optimal matching between two sets of elements. Developed by Harold Kuhn in 1955, it originally applied to workforce assignment problems but has since become foundational in data association tasks across fields like computer vision and multi-object tracking. In 3D tracking applications, the Hungarian algorithm plays a critical role in associating detected objects across frames, enabling robust tracking in dynamic environments. Some of the challenges of 3D data association are listed below:

- **Multiple Detections and Occlusions:** When tracking multiple objects in a scene, objects may enter or leave the field of view, leading to partial observations or occlusions that complicate consistent tracking.
- **Dynamic Environments:** Objects may exhibit irregular or non-linear motion, increasing the complexity of associating detections with past observations.
- **Measurement Noise:** Sensor noise and environmental conditions, such as lighting in computer vision systems or interference in radar systems, introduce uncertainty in the positions of detected objects.

The Hungarian algorithm addresses these challenges by efficiently determining the optimal assignment between objects in consecutive frames, minimizing mismatches and improving overall tracking accuracy. The specific challenges are addressed below:

- **Minimized Cost for Accurate Matching:** By minimizing the overall association cost, the algorithm reduces mismatches, ensuring that objects are consistently and accurately tracked.
- **Efficient Handling of Occlusions and Reappearances:** The algorithm is capable of handling cases where objects are temporarily occluded and then reappear, maintaining object identity over time.
- **Scalability and Real-Time Compatibility:** The Hungarian algorithm's polynomial-time complexity makes it efficient enough for real-time applications, which is critical in scenarios like autonomous driving or AR/VR, where computational resources and processing time are limited.

In summary, the Hungarian algorithm's role in 3D tracking lies in its ability to provide efficient, optimal data association in multi-object settings, addressing challenges in occlusions, noise, and dynamic environments. Its application, especially when integrated with predictive filters like the Kalman filter, enables robust and accurate tracking in real-time, making it indispensable in multi-object tracking systems.

C. You Only Look Once (YOLO)

You Only Look Once (YOLO) [20] is a real-time object detection algorithm initially proposed by Joseph Redmon et al. in 2015 [19]. The core innovation of YOLO lies in its ability to detect and localize multiple objects in an image or frame by performing a single, unified analysis. Unlike traditional object detectors that rely on region proposals or multiple passes, YOLO treats object detection as a single regression problem, predicting bounding boxes and class probabilities simultaneously. In the context of 3D tracking, YOLO is widely used for its high-speed detection capabilities, essential for applications that require real-time tracking and analysis.

YOLO's ability to detect objects in real-time makes it highly useful in 3D tracking pipelines, particularly for applications like autonomous driving, unmanned aerial vehicles (UAV) navigation, and augmented reality. Typically, YOLO detects objects in 2D image frames from sensors (e.g., cameras or LiDAR) and, when combined with depth information from stereo cameras or other sensors, provides approximate 3D positions of objects. The workflow of the YOLO model is given below:

- **Object Detection:** YOLO detects objects within each image frame, producing 2D bounding boxes along with confidence scores and class labels.
- **Depth Estimation and Position Mapping:** For 3D tracking, depth information is obtained through stereo vision, LiDAR, or depth sensors. This depth data enables YOLO's 2D detections to be mapped into 3D coordinates, allowing approximate localization in 3D space.
- **Data Association and Tracking:** Once YOLO has detected objects and their corresponding 3D coordinates, data association techniques (e.g., Hungarian algorithm) are used to match these detections across consecutive frames. Combining YOLO with Kalman filters or other prediction algorithms allows for smoother and more robust tracking.

The main advantages of using YOLO in 3D tracking are given below:

- **Real-Time Detection:** YOLO's single-pass detection structure makes it one of the fastest object detection algorithms, which is essential for real-time tracking in applications like autonomous navigation and AR/VR.
- **Unified Detection and Classification:** YOLO simultaneously detects and classifies objects in each frame, enabling streamlined data association when tracking objects across frames.
- **Adaptability to Complex Scenes:** YOLO can detect multiple object types within a single frame, allowing it to track diverse objects in complex 3D environments, from vehicles and pedestrians to small objects like packages and equipment.

To perform 3D it is best to combine the detection ability of YOLO with other algorithms such as the Kalman filter and the Hungarian Algorithm. Further details are given below:

- **YOLO and Kalman Filter:** YOLO provides accurate detections, while the Kalman filter predicts object trajectories, helping smooth out positional estimates over time and reducing jitter in 3D tracking.
- **YOLO and the Hungarian Algorithm:** In multi-object tracking, the Hungarian algorithm helps maintain object identities across frames by associating YOLO's detections in consecutive frames, particularly useful when tracking multiple, similar objects.

In summary, YOLO's fast and efficient detection capabilities make it highly suitable for 3D tracking, especially in scenarios demanding real-time performance. Its adaptability, combined with methods for depth estimation and

data association, ensures YOLO's effectiveness in a wide array of 3D tracking applications.

D. Image Embeddings

Image embeddings [13] are numerical representations of images generated by deep learning models that capture the essential visual features of an image in a high-dimensional vector space. These embeddings serve as compact and informative summaries of images, making them useful for various tasks, including image retrieval, clustering and classification. By representing an image as a vector, embeddings allow for comparisons between images based on their visual similarity, which is measured as the distance between their vectors. The key concepts of image embeddings are given below:

- **Feature Extraction:** A neural network, typically a convolutional neural network (CNN) or transformer-based model, processes an image to extract meaningful features, such as shapes, colors, and textures. Layers in these networks gradually refine the raw pixel data into high-level abstractions.
- **Dimensionality Reduction:** The complex data in an image is reduced to a lower-dimensional vector, capturing essential characteristics while discarding irrelevant details. Common embedding dimensions range from a few hundred to a few thousand.
- **Similarity Measurement:** Embedding vectors enable comparisons between images based on their proximity in the embedding space. Similar images have closer vectors, allowing for applications like similarity-based search and duplicate detection.

Overall, image embeddings transform complex visual data into a structured format that machines can process efficiently, supporting a range of computer vision tasks and making it easier to work with images in high-dimensional spaces.

E. Siamese Neural Networks

Siamese Neural Networks (SNNs) [14] are a type of neural network architecture specifically designed to compare two inputs by learning their similarity. Instead of generating a single output, SNNs use two (or more) identical subnetworks with shared weights to process the input pairs independently and then compare their outputs. This design makes SNNs ideal for tasks involving similarity detection, such as face recognition, signature verification, and one-shot learning. The key concepts of a Siamese neural network are given below:

- **Twin Networks:** SNNs consist of two identical subnetworks, typically CNNs for image inputs, which

process the two inputs in parallel. These subnetworks share weights, ensuring they learn similar representations regardless of the input order.

- **Distance Metric:** After the subnetworks process each input, their outputs are compared using a distance function, such as Euclidean or cosine distance, to measure the similarity between the two embeddings generated.
- **Contrastive or Triplet Loss:** SNNs are trained with loss functions that encourage similar inputs to have closer embeddings and dissimilar inputs to have more distant embeddings. Contrastive loss minimizes the distance for similar pairs, while triplet loss uses an anchor-positive-negative triplet setup for improved discrimination.

Siamese Neural Networks are powerful for tasks that require comparison and verification, as they excel at learning meaningful relationships between inputs based on similarity rather than simply classifying or labelling data.

III. METHODOLOGY

This section covers how the various elements are implemented and how the system functions. The siamese model and image embedding can be interchanged within the system as they are being compared for effectiveness at reidentification. The image embedding model that is used is provided by Hugging Face

A. Dataset

The dataset used in this paper consists of images extracted from the Wildtrack dataset as well as various real-world images extracted from security cameras. Further information about the Wildtrack dataset can be viewed at <https://www.kaggle.com/datasets/aryashah2k/large-scale-multicamera-detection-dataset/data>. For both of these sets of data, cleaning needed to be done first - especially the security camera set as there were large amounts of video where nothing occurred in the frame. This was even more true for the infrared mode of the security cameras. The diagram below gives an idea of how the security camera system is set up.



Fig. 1: Security Camera System

The colour triangles in Figure 1 represent the three cameras that make up the system and their view of the road which is represented by the grey line. The red dot represents an object moving along the road. The red dot can either be moving from left to right or from right to left.

B. Siamese Neural Network

For the the construct of the Siamese Neural Network used in this paper the following steps are followed:

- 1) We start by importing the necessary Python libraries from TensorFlow and NumPy.
- 2) The next step is defining our neural network structure in this paper. The network is constructed with two sets where a single set is made of a convolutional layer with the "relu" activation function, followed by a max-pooling layer and finally a dropout layer. After the two sets of layers are constructed the output layer is constructed.
- 3) The next step is to load the dataset that will be used for training. In our case, we have three different sets that need to be trained on their own network.
- 4) After the chosen dataset is loaded it needs to have its pixel values scaled to [0,1] and split into its training and test sets.
- 5) Next for both the training and test split pairs of images are constructed where a pair either represents the same object or not. This is what the network will be trained on.
- 6) Next we construct our Siamese Neural Network from the structure we made earlier.
- 7) Next we train the neural network.
- 8) Finally we save a serialised version of our network for use in our image detection pipeline.

C. Object Tracking Pipeline

For the construction of the object detection and tracking system the following steps are followed:

- 1) We start by importing the necessary Python libraries
- 2) Next we start by breaking down the stream from each of the cameras into the individual frames.
- 3) Then we construct the Obj class that is used to store the information for an object. The information stored is the "object id", "previous X location", "previous Y location", "Current X location", "Current Y location", "the Velocity" and finally the "image embedding".
- 4) Next we define a function to take an image and embed it. This is done using methods from the transformers library provided from Hugging Face

as well as the pre-trained model "Convnext-small-224" provided by Facebook. This is done only if image embedding is the method that is being used. Not performed in the case of using the Siamese network. In the case of the Siamese network only the original bounding box is stored. For comparison to allow reidentification.

- 5) The next step is to define the function to calculate the similarity of two embeddings.
- 6) Next we define a function to find the object that has the most similar embedding in a list of objects to a given object.
- 7) Next we define a function to determine the area of intersection between two objects given the top left corner value and the bottom right corner value.
- 8) Next we define the Hungarian algorithm that was introduced in Section II.
- 9) Now we put it all together by looping through the frames of the cameras to simulate real-time data where on each frame the objects are detected and identified, and when an object leaves a camera's view they are added to an expected list for the next camera in the system. In the case where the cameras overlap - when an object crosses the so-called hot zone which is the area where the cameras overlap - the camera where the object is entering will simply be reidentified as the first camera will inform the second which object just crossed the hot zone. In the case where there is a gap between the cameras when the object leaves the first camera, it is added to an expected list of objects for the next camera. When a new object is detected coming from the direction that the object was last seen at, the image is reidentified using either the Siamese network that we created and trained or using the embeddings - depending on the current configuration of the system.

D. Motion Model

The motion model used in this paper is based on the Equations of Motion. The specific equation used is.

$$r_1 = r_0 + v_0 t + \frac{1}{2} a t^2 \quad (1)$$

Where

- r_1 is the future predicted position
- r_0 is the current position
- v_0 is the current velocity
- t is the change in time
- a is the acceleration

For this paper the change in time will always be one as we are predicting the position between frames. The steps of the motion model are as follows:

- 1) When an object is detected for the second time within the same camera. The velocity is calculated. This is repeated for all following detection of that object within that camera.
- 2) The acceleration is calculated after the second detection similar to the velocity with a initial velocity of 0. The acceleration is updated after each detection similar to the velocity.
- 3) Using the calculated acceleration and velocity and Equation 1, a predicted value for the x and y position of the object centre is calculated.

IV. IMPLEMENTATION

This section provides an overview of the libraries used to implement the methods discussed in Sections II and III.

A. Basic summary of libraries used

The TensorFlow Python library [1] was used for all neural network construction. The NumPy Python library [10] was used for matrix and array construction and interaction. The YOLO Python library from Ultralytics [20] was used for all object detection.

B. Neural Network Construction Overview

The Siamese model is custom built using packages from the TensorFlow [1] library. Each individual network that comprise the twin networks of the Siamese network are constructed using the convolutional layer, a RELU layer, a pooling layer and a dropout layer.

C. Matrix and Array Construction Overview

All images that are used in this paper are converted to a NumPy array when they are being interacted with. These arrays are then used in the analysis and also what is supplied to the image embedding method used to reidentify objects, as well as the Siamese network and Hungarian algorithm.

D. Object Detection Overview

Each frame is extracted from the camera system and is then passed through YOLO to identify all objects present in each frame. From those individual frames only the objects that are detected as a person are extracted by the system.

V. RESULTS AND DISCUSSION

In this section, we examine an example of the system running and compare the results of using the different re-identification techniques.

A. Example of Detections

The figure 2 below shows an example of the YOLO detection.

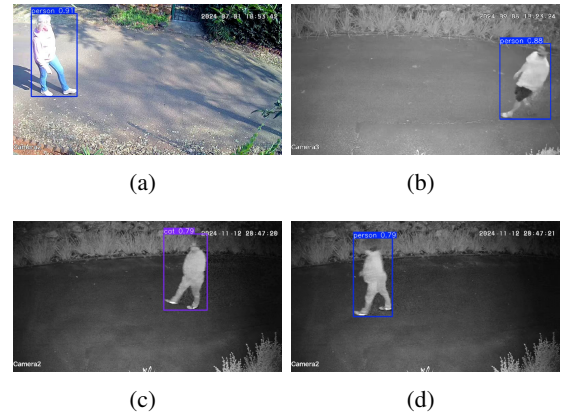


Fig. 2: (a) Daylight Detection (b) Infrared Detection at moderate darkness (c) Incorrect Infrared Detection (as a cat) at extreme darkness (d) Correct Infrared Detection at extreme darkness

Figure 2 shows YOLO's visual representation of the bounding box and the accuracy it finds for all objects in the current frame. Figure 2 (b), (c) and (d) also shows that YOLO works for multi-spectrum images, the infrared spectrum in this case. However, Figure 2 (c) does indicate that although YOLO does not struggle in finding objects in infrared spectrum, the amount of light does influence the YOLO model's ability to identify the category an object belongs too, *id est* confusing a person with a cat.

B. Example of Reidentification of Objects

Let us now look at an example where the first detection in every camera and the label for the object will be shown. The first object to enter the system is shown in Figure 3.



Fig. 3: (a) Object0 first appearance Camera 01 (b) Object1 first appearance Camera 01

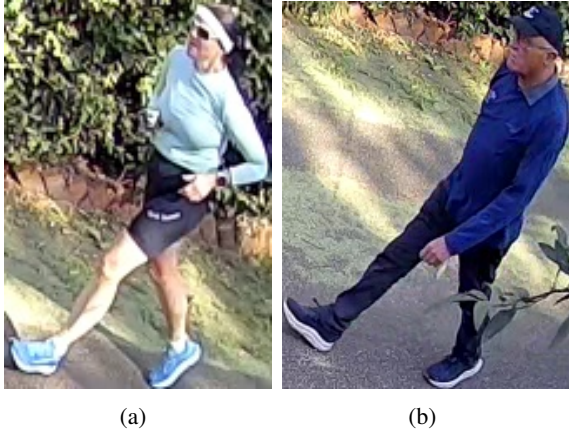


Fig. 4: (a) Object1 first appearance Camera 02 (b) Object0 first appearance Camera 02



Fig. 5: (a) Object1 first appearance Camera 03 (b) Object0 first appearance Camera 03

The cropped-out object from the initial embedding that was calculated and stored for Object0 and Object1 are depicted. For reference, Object0 is the first to enter Camera 01. However, the first object detected in Camera 02 is Object1 (Figure 4 a). Figure 4 (b) shows the initial embedding captured upon Object1's entry into Camera 02. This embedding is compared using a reidentification technique be it the Siamese network or image embeddings with the two stored embeddings captured from Camera 01. Hence it is determined that this is Object1 - not Object0. The limitation of image embedding however, is that during the calculation of the embedding, the system slows down by a large degree. This limitation is however not present in the Siamese network. The Siamese network does suffer from having a small dataset of samples for this system. This problem would disappear as more data are gathered naturally from the system.

In the example, Object1 leaves Camera 02 before Object0 enters Camera 02, hence Object1 is detected in Camera 03 before Object0 has enter Camera 02. Given that Camera 02 and Camera 03 don't have a blind spot between them, when Object1 leaves Camera 02 the system is aware of when Object1 transitions from Camera 02 to Camera 03, hence there is no need to recalculate the embeddings. The reason Object0 has not been detected again, is due to the gap between the Camera 01 and Camera 02 as can be seen in Figure 1 which has allowed Object1 to create such a gap. The system of cameras is not isolated, hence new objects could enter the system whenever.

At the time when Object0 enters Camera 02, Object1 is still in the system. A similarity check is performed by the system to determine whether the new detection in Camera 02 is Object0, Object1 or a new object. The system does determine that the new detection in Camera 02 is Object0. The hand-off of objects between Camera 02 and Camera 03 is the same for Object0 as described above for Object1. The first detection of Object0 in Camera 03 can be seen in Figure 5 (b). The example is for the normal colour spectrum, but the system functions the same with an infrared example. The object detection does however differ as discussed in Subsection A of the Results.

A different Siamese Neural Network must be used as it needs to have been trained on infrared images. For the infrared case, the image embedding strategy would definitely be the better identification technique with how rare occurrences of infrared detection are in this system. However, if a different infrared system was created with more data, the Siamese Neural Network would be the better option.

C. Reidentification model Comparison

To examine the strengths of the reidentification methods, we compare the frames per second that each can process. The results can be seen in Table 1, listing the average fps (frames per second) and variance that each reidentification method achieved over 30 independent runs.

Method	Avg fps	Variance
Image embedding	11.44883	3.3085
Siamese network	20.17136	0.1369

TABLE I: 10-fold Mean test accuracies

From Table 1 it seems that the Siamese method is the superior method to image embedding as its fps is nearly twice that of image embedding while also having a lower variance. To determine whether there is a statistical difference between the two reidentification methods a t-test is used. The t-test with a α value of 0.05 determined that there is certainly a statistical difference between these two methods. The p-value determined using the test was found to be $2.105577e - 21$. This is vastly smaller than the α value.

D. Multi-spectrum aspect

From the results thus far, it is clear that the Siamese network outperforms image embedding. However, this does come with the caveat of there being sufficient data available to train the Siamese network. This flaw was evident in the system set-up for this project as the amount of data available for the infrared spectrum was very limited. This lead to the Siamese model, trained for the infrared spectrum, to perform worse with regards to the networks ability to accurately determine the reidentification for objects. This shows that in an ideal situation, the Siamese network is the superior method but image embedding is more robust in the case of sparse data.

E. Prediction Model

The motion model discussed in Section III results in semi-accurate prediction as can be seen in Figure 6.



Fig. 6: The results of Prediction Model

Figure 6 shows the results of the prediction model for Object1 in Camera 01 from the example mentioned

earlier. The red dots indicate the predicted centre of the object and the green dots indicate the actual centre of the object. One of the factors that leads to the disparities between the real location and the predicted location, is that the centre of an object is used as the metric for the prediction; due to the changes in the size of the bounding box the centre shifts. Meaning that the point in the detected object that the centre represents, is not consistent between different detections. However, even with this complication the predictions are not off by a large degree.

F. Use Case

The system built for this paper has a wide variety of real-world applications. For example, a general use case could be tracking the movement of people through a store to determine which isle and areas are your most visited hence determining where the hot and cold spots within your store are. Another example - tracking vehicles through a neighbourhood to determine which roads are most used and therefore would require more maintenance - but also provide valuable statistics to sell advertisement along these roads.

VI. CONCLUSION

We have shown that the algorithms used in this paper perform well and are applicable to both the normal day-light and the infrared light spectrums. It was also found that image embedding is a good technique to use when there is not available time or enough data to train a Siamese Neural Network. Future projects of this kind could include implementing multiple angles and extending the system to detect vehicles - not only pedestrians.

REFERENCES

- [1] Martín Abadi, Ashish Agarwal, Paul Barham, et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: <https://www.tensorflow.org/>.
- [2] Temitope Ibrahim Amosa, Patrick Sebastian, Lila Iznita Izhar, Oladimeji Ibrahim, Lukman Shehu Ayinla, Abdulrahman Abdullah Bahashwan, Abubakar Bala, and Yau Alhaji Samaila. “Multi-Camera Multi-Object Tracking: A Review of Current Trends and Future Advances”. In: *Neurocomputing* 552 (2023), p. 126558. DOI: [10.1016/j.neucom.2023.126558](https://doi.org/10.1016/j.neucom.2023.126558). URL: <https://doi.org/10.1016/j.neucom.2023.126558>.
- [3] Vaibhav Bansal, Gian Luca Foresti, and Niki Martinel. “Where Did I See It? Object Instance Re-Identification with Attention”. In: *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*. 2021, pp. 298–306. DOI: [10.1109/ICCVW54120.2021.00038](https://doi.org/10.1109/ICCVW54120.2021.00038).
- [4] Calib Jonas Buckton. “Multi-spectral Object Tracking and Prediction of Kinematic Quantities”. In: PP (Dec. 2021).
- [5] Tatjana Chavdarova, Pierre Baqué, Stéphane Bouquet, Andrii Maksai, Cijo Jose, Louis Lettry, Pascal Fua, Luc Van Gool, and François Fleuret. *The WILDTRACK Multi-Camera Person Dataset*. 2017. arXiv: [1707.09299](https://arxiv.org/abs/1707.09299) [cs.CV]. URL: <https://arxiv.org/abs/1707.09299>.
- [6] Cheng-Yao Chen and Wayne Wolf. “Background modeling and object tracking using multi-spectral sensors”. In: *Proceedings of the 4th ACM International Workshop on Video Surveillance and Sensor Networks*. VSSN '06. Santa Barbara, California, USA: Association for Computing Machinery, 2006, pp. 27–34. ISBN: 1595934960. DOI: [10.1145/1178782.1178788](https://doi.org/10.1145/1178782.1178788). URL: <https://doi.org/10.1145/1178782.1178788>.
- [7] Minh Cho and Euntai Kim. “3D LiDAR Multi-Object Tracking with Short-Term and Long-Term Multi-Level Associations”. In: *Remote Sensing* 15 (Nov. 2023), p. 5486. DOI: [10.3390/rs15235486](https://doi.org/10.3390/rs15235486).
- [8] Ricardo Dias, Bernardo Cunha, Eduardo Sousa, José Luís Azevedo, João Silva, Filipe Amaral, and Nuno Lau. “Real-time multi-object tracking on highly dynamic environments”. In: *2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. 2017, pp. 178–183. DOI: [10.1109/ICARSC.2017.7964072](https://doi.org/10.1109/ICARSC.2017.7964072).
- [9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: *CoRR* abs/2010.11929 (2020). arXiv: [2010.11929](https://arxiv.org/abs/2010.11929). URL: <https://arxiv.org/abs/2010.11929>.
- [10] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, et al. “Array programming with NumPy”. In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: [10.1038/s41586-020-2649-2](https://doi.org/10.1038/s41586-020-2649-2). URL: <https://doi.org/10.1038/s41586-020-2649-2>.
- [11] Hou-Ning Hu, Yung-Hsu Yang, Tobias Fischer, Trevor Darrell, Fisher Yu, and Min Sun. “Monocular Quasi-Dense 3D Object Tracking”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PP (Apr. 2022), pp. 1–1. DOI: [10.1109/TPAMI.2022.3168781](https://doi.org/10.1109/TPAMI.2022.3168781).
- [12] Rudolph Emil Kalman. “A New Approach to Linear Filtering and Prediction Problems”. In: *Transactions of the ASME—Journal of Basic Engineering* 82.Series D (1960), pp. 35–45.
- [13] Valentin Khrulkov, Leyla Mirvakhabova, Evgeniya Ustinova, Ivan V. Oseledets, and Victor S. Lempitsky. “Hyperbolic Image Embeddings”. In: *CoRR* abs/1904.02239 (2019). arXiv: [1904.02239](https://arxiv.org/abs/1904.02239). URL: [http://arxiv.org/abs/1904.02239](https://arxiv.org/abs/1904.02239).
- [14] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. “Siamese Neural Networks for One-shot Image Recognition”. In: 2015.
- [15] Harold W. Kuhn. “The Hungarian Method for the assignment problem”. In: *Naval Research Logistics Quarterly* 2 (1955), pp. 83–97.
- [16] Xiangyuan Lan, Zifei Yang, Wei Zhang, and Pong C. Yuen. “Spatial-temporal Regularized Multimodality Correlation Filters for Tracking with Redetection”. In: *ACM Trans. Multimedia Comput. Commun. Appl.* 17.2 (2021). ISSN: 1551-6857. DOI: [10.1145/3430257](https://doi.org/10.1145/3430257). URL: <https://doi.org/10.1145/3430257>.
- [17] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: [10.1109/5.726791](https://doi.org/10.1109/5.726791).
- [18] Zhuohao Li, Fandi Gou, Qixin De, Leqi Ding, Yuanhang Zhang, and Yunze Cai. *RealNet: Combining Optimized Object Detection with Information Fusion Depth Estimation Co-Design Method on IoT*. Apr. 2022.
- [19] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. “You Only Look Once: Unified, Real-Time Object Detection”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition*

- tion (CVPR). 2016, pp. 779–788. DOI: [10.1109/CVPR.2016.91](https://doi.org/10.1109/CVPR.2016.91).
- [20] Ao Wang, Hui Chen, and et al. Lihao Liu. “YOLOv10: Real-Time End-to-End Object Detection”. In: *arXiv preprint arXiv:2405.14458* (2024).
 - [21] Zhongdao Wang, Liang Zheng, Yixuan Liu, and Shengjin Wang. “Towards Real-Time Multi-Object Tracking”. In: *CoRR* abs/1909.12605 (2019). arXiv: [1909.12605](https://arxiv.org/abs/1909.12605). URL: <http://arxiv.org/abs/1909.12605>.
 - [22] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. “Simple online and realtime tracking with a deep association metric”. In: *2017 IEEE International Conference on Image Processing (ICIP)*. 2017, pp. 3645–3649. DOI: [10.1109/ICIP.2017.8296962](https://doi.org/10.1109/ICIP.2017.8296962).