Human Detection, Classification & Tracking in Context of Transit Systems

Chigozie Ofodike[1] and Christoph Mertz[2]

***Abstract*— With the advent and popularity of algorithms capable of detection-based tracking (DBT), one of its growing applications is in human detection and tracking (HDT). Leading research in HDT can be seen in surveillance systems, anomaly detection (e.g fall detection for senior citizens), and recently, social distance monitoring. In this paper, we present an application of contemporary HDT algorithms, on a real-time and ubiquitous entity–the mid-tech public transit bus system. All forms of DBT follow two innate steps: first, object detection, then association of the current object with the previous object. In the case of human detection, every instance of a detected human is then analyzed. In our project, we want to perform visual tracking from the transit bus. Each implementation is done by aggregating data (mainly pictorial) from cameras mounted on a bus with the Robotics Operating System (ROS) acting as the architecture supporting both the bus and the server structures. Our proposed system will allow for technological automations and implementations for human-specific observations.**

***Index Terms*— keywords, Intelligent Transportation Systems, Surveillance Robotic Systems, Software, Middle-ware and Programming Environments, Computer Vision for Transportation**

I. INTRODUCTION

With its wide use cases, algorithms involved in Human Detection and Tracking (HDT) have become relevant in recent years. Applications of these algorithms can be seen in surveillance systems, self-driving cars and even anomalous action detection in environments (i.e. a fall or a vandalism) [1]. The core steps of these algorithms are: informative region selection, feature extraction then classification (specifying person) [2]. Beyond these stages, most of the computation overhead occurs during the tracking stage [3]. Understanding that humans tend to be erratic, and often without a standardized shape, movement pattern or appearance, this overhead becomes more evident in cluttered environments or scenes with dense crowds [1], [4], [5]. Fortunately, many state-of-the-art (SOA) paradigms have suitable and efficient means to handle some of these resulting issues.

Our implementation is currently done on a single bus provided by Freedom Transit, a bus transportation service in Pennsylvania. Currently, in order for their analysts to review the recorded video of a bus' route, they must wait until the end of the day, when bus drivers are completely done with their respective routes. Then and only then are they able to gain access to it. After which analysts must sit and manually annotate the highly-repetitive and long streams of videos. This process is not only tediously time-consuming, it is prone to errors and it causes delays for timely information extraction. In this paper, we propose an implementation of real-time HDT to assist this system. Our proposed system will involve object detection and classification, to train a model to detect pedestrians. A classification mechanism to differentiate pedestrians from passengers either exiting or entering the bus.

The structure of this paper is as follows: In section II, we present our background study, showing related works, SOA object detection and tracking architectures. In section III, we present our system overview and our contributions to this system. Section IV will be our results and evaluations. V will be our conclusions and finally, VI will be our future work and discussions.

II. BACKGROUND STUDY

A. Related works

1) General Object Detection: Research in object detection and tracking (ODT) architectures have grown prevalent in recent times. Applications of ODT can be seen in medical imaging, automated robotics, image recognition and even surveillance systems [6], [7]. As mentioned in [1], [5] traditional object detection works by informative region selection, feature extraction, and classification. Traditional region selection is done with a sliding window approach. This method works by taking exhaustive sliding rectangular "patches" of fixed width and height for each image. Feature extraction then happens on each derived patch. After which a classifier is used to distinguish between objects in each frame. Due to the exhaustive nature required with the sliding windows, this traditional method is ineffective with real-time analysis [5], [8]. Nowadays, with the prevalence and utilization of Convolution Neural Networks (CNN) and deeply trained models, detections algorithms can occur at a much faster rate. We briefly discuss these SOA models in a later section.

2) Object Detection on Embedded Devices: Since these architectures need to be deployed to put to use, another popular research area is real-time object detection on embedded devices. Applications of object detection on embedded devices could be seen on autonomous vehicles, robotics, and C. Ye's BusEdge system [6]. Despite the recent successes of these algorithms, an issue common with the applications of object detection and tracking on embedded systems is the limited resources of these micro-systems. It becomes a question of which object detection architectures give us

[1]Chigozie Ofodike is with Kean University, Union NJ, USA. Ofodikch@kean.edu

[2]Christoph Mertz with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA cmertz@andrew.cmu.edu

the best tradeoff between speed and accuracy, while still effectively running on much lighter processors [9]. With the combination of significantly smaller processors and a computationally expensive algorithm, this is a difficult task. For one thing, when analyzing real-time data, [6] points out that most are unusable and very redundant. To that avail, there have been several successes that circumvent this issue. For example, Mobileye is a self-driving car company that implements a Road Experience Management system (REM). The company claims this system works by automatically uploading and processes anonymized data from cars already running its technology onboard [7]. Another company relevant in this space is RoadBotics. RoadBotics allows users to upload road-specific data to their server, then their deployed model handles the classification and analysis [10]. Specific to our needs, another leading approach facilitating this complex problem is NavLab's BusEdge system [6]. It employs an edge computing paradigm and an application called AutoDetectron. Being that our system is an extension of his, we talk more about this in our "System Overview" section.

3) Pedestrian Detection and Tracking: Being that our project is fixated on how humans interact in relation to the transit system, our focus is strongly on pedestrian detection, tracking and observation. With the vast applications of human specific observations of object detections, there have been a multitude of implementations of SOA algorithms in this area. Zhang et al. [11] propose a method based on Tiny Yolo, training a model to detect the upper body of passengers entering/exiting a bus. Valastin et al. [12] show their different approaches to detection, tracking and crowd counting of pedestrians getting on and off a Metropolitan Train. They carry out their experiments in their reshapeable lab Pedestrian Accessibility Movement Environment Laboratory (PAMELA). This re-shapeable feature allows their lab to properly train pedestrian models no matter how rare, common, or messy of an event it is. They are able to handle problems we meet in our implementation such as properly detecting passenger flow (i.e. they are able to differentiate passengers entering from those exiting) and crowd counting of an active and crowded scene.

B. State of the Art Object Detection Architectures

1) Faster RCNN: Originally proposed by Ren et al. [13] RCNN, these architectures mainly consist of a layer of convolution neural networks (CNN) and Region Proposal Network (RPN). CNN is trained to extract appropriate features from the image, in this case features that appropriately describe humans. The RPN is a small neural network sliding on the last feature map of the convolution layer, predicting the existence of an object and the bounding box if an object is detected. The massive increase in analysis speed from 10 milliseconds per image to 2 milliseconds per image is heavily credited to this layer.

2) YOLO (You Only Look Once): Redmon et al. [14] propose a regression approach to object detection that requires only a single look at an image for object detection. It consists of 24 convolutional layers and two fully connected layers and as the name suggests, it requires only one single forward propagation through the layers to detect objects. When compared to the architectures of RCNN, it tends to make more localization errors, but false positives are far less likely. In terms of speed of processing, YOLO's base model easily outperforms the already fast Faster RCNN–processing at 45 frames per second (fps) [15]. YOLO like RCNN comes with other versions, with its fast version processing at more than 150 fps. With such a massive processing rate, it is very suitable for live video processing with less than 25 milliseconds of latency [15].

3) H-YOLO: A Single-Shot Ship Detection Approach Based on Region of Interest Pre-selected Network: Although about ships, Tang et al. [16]. proposal of a single-shot detector on the pre-selected region of interest was the starting point of our passenger classifier. Using hue, saturation and value color space operations and a one-shot detector, they were able to extract pre-processed regions of interest at close to real-time.

4) Single Shot MultiBox Detector (SSD): Proposed by Liu et al. [17], SSD is a competing object detection model that works with a single phase analysis to detect multiple objects within the image. The SSD network is built on the VGG-16 model, where the feature map is extracted without the need of the bounding box proposals like that of RCNN. This map is then processed through six progressively decreasing convolution filters (the multi-box), generating. The use of multiple levels of filters allows about 8732 detections per object (class). The final layer, non-max suppression layer, eliminates the overlapping box by performing a bounding box regression effectively leaving the calculated final box with the highest overlap [17]. SSD's strength lies with its balance of ease of training speed and accuracy–being faster than both Faster RCNN and the base YOLO model, and more accurate than other single-stage methods.

C. State of the Art Tracking Models

1) Simple Online and Real-time Tracking with a Deep Association (Deep SORT): Deep SORT is the successor of SORT [18]. SORT is a high-performing two-stage tracking-by-detection framework that performs Kalman filtering in image space and data association using the Hungarian method. Put forth by Wojke et al. [18], Deep SORT builds upon its predecessor by adopting a deep association metric with recursive Kalman filtering on frame-by-frame data [5]. Although slower than its predecessor, the added association step drastically reduces the occurrence of identity switches among detected instances [18].

2) Joint Detection and Embedding (JDE): Proposed by Wang et al., unlike the two-stage tracking style of Deep SORT, JDE integrates the detector and embedding model into a single network. The combination of both stages removes the need for an additional layer of computation, therefore reducing the inference time [19]. Unfortunately, with this combination, these methods tend to be significantly less accurate than 2-stage methods, although capable of achieving near video rate inference [1].

3) FairMOT: A one-stage tracker that aims to balance speed and accuracy. Using an anchor-free detector, a heatmap, then a multiple-level data association step involving bounding box intersection over union, re-id features and Kalman filtering [20]. FairMOT's advantage comes from understanding that previous SOA's re-identification system is poor, due to its detection module being heavily favored over its re-id module. This work proposed by Zhang et al., aims to balance both modules. By implementing a multi-layer feature aggregation framework, its re-id module improves significantly.

III. System Overview

Our detection pipeline will be deployed on the Gabriel BusEdge system as proposed and explained by C. Ye. [6]. The system's major components are hardware, early-discard filters, cognitive engines and finally sinks. Hardware on the bus are multiple wide angle cameras mounted at different positions on the bus, interior cameras, GPS, network antenna and an accelerometer (see Figures 1 and 2). The early-discard filters are Ye's lightweight preprocessing mechanism, and we choose SSD with MobileNet. This allows ad hoc analysis instead of analysis on highly-repetitive data effectively promoting scalability. Cognitive engines are the more computationally heavy computer vision models that handle and analyze distilled data from the bus, YOLO is our choice here. Finally the sinks, which represent the final component used for data analytics and visualizations. Our choice here is Christensen's proposed LiveMap system [21]. The complete system uses Robotics Operating System (ROS) as its base architecture. See [6] for a more detailed and thorough description for the Gabriel BusEdge system.

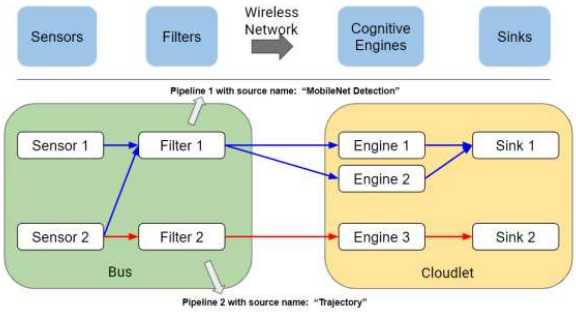


Fig. 1: An abstraction of C Ye's BusEdge pipeline.

Fig. 2: Pictures of the Hardware. (From left to right: bus computer; exterior camera; interior camera; GPS and network antenna.)

A. Our Contributions

Our implementation is centered around aggregating data for human activity around the transit system. Although at the beginning stages, we implement a human detection model, then a classification module capable of differentiating pedestrians around from those boarding on a specific transit route. Being that humans are the focal point of public transit, this creates a solid starting point for future automations.

B. Proposed Methodology for our Pedestrian Detection and Passenger Classification.

1) Grab and read a frame.
2) Apply a fine-tuned object detection model to each individual frame. To only detect people, we simply discard the information of every other class.
3) Get and store bounding boxes, scores and labels of each instance of each detected human.
4) Select only pedestrians above a .70 confidence score– this way we eliminate false positives such as mannequins and road signs at a distance.
5) Instantiate a passenger counter.
6) Create an invisible bounding box (we call this our boarding zone) where passengers must enter or exit.
7) For each properly identified pedestrian instance, if the instance crosses the boarding zone, we compute the intersection over union (IoU) of our zone against the detected pedestrian.
8) If the resulting confidence score is greater than .50, this person instance is treated as a passenger.

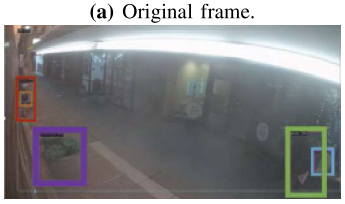
IV. Results and Evaluations

1) Results: Although promising, there are still some challenges with our pipeline. First off, as shown in Fig 3, there tends to be some mislabelling of pedestrian instances. Instances such as this, and accidental mannequin selections occur at a confidence threshold of .75. Increasing the threshold further would cause the model to ignore instances in darkened scenes (such as that in Fig 4). Another common issue that came up are pedestrian instances missed because of slight occlusions (shown in Fig 5). This specific instance is barely covered by the obstructing sign, yet it was completely ignored. This same instance in the frame before commanded a .90 confidence score. This can easily be fixed by either implementing a SOA tracker, allowing preservation of the identity of each instance between each frame change, even behind partial obstructions or the re-identification of instances even through complete obstruction.

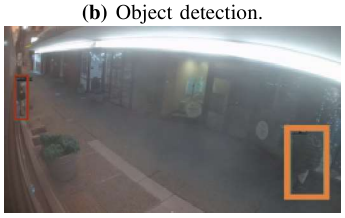
Fig. 3: Sign to the extreme right mislabelled as a person.

2) Evaluations: Evaluating our logic on unseen data, we see some issues. Our pedestrian detector seems to get all pedestrians in a given scene, but it fails to differentiate between a reflection and the actual passenger as seen in Fig. 6b. This was an unanticipated event, as it is the first time such a detection occurred. Being that the passenger

(a) Original frame.



(b) Object detection.



(c) Fine-tuning for pedestrians.

(d) Passenger specification.

Fig. 4: Successful pedestrian detector and passenger selection.

classification module only cares about the instance in the invisible boarding zone, this still seems like an effective method of passenger classification.

V. Conclusions

In this paper, we proposed an addition to Freedom Transit's current analysis pipeline. With the help of SOA algorithms, and C Ye's BusEgde system as the architecture behind this, we were able to effectively detect humans around specific bus routes. The research and work done here provides a foundation to assist Freedom transit with their data analysis processes.

VI. Future and Discussions

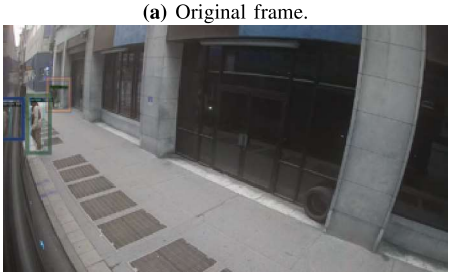
Future plans will be to use already existing benchmarks such as those provided by [22], [11] and [23]. [22] works well with detecting occluded instances in urban areas, whereas [11] is a significantly more diverse pedestrian dataset. [23] claims to have a strong generalization ability. Evaluation of these datasets, recommended training paradigms and more were presented well by Hasan et al. [2]. Another implementation we plan on is for people with disabilities. Rarely are they adequately represented in popular pedestrian datasets. Then, a tracker for all instances. Giving our system the ability to tell detected humans apart, effectively establishing metrics for possible analysis. A counter-

(a) Pedestrian on the extreme left will be missed in the next frame.

(b) Instance missed due to occlusion.

Fig. 5: Interesting, yet correctable misses.

(a) Original frame.



(b) Person detection evaluation.

(c) Passenger specification evaluation.

Fig. 6: Evaluation of pedestrian detector and passenger selection.

intuitive, yet interesting notion put forth by [2] is that general object detection models tend to work better than current SOA pedestrian-specific detectors on new and population-dense scenes. We will analyze and compare SOA pedestrian-detectors, SOA general object detection models, and our pedestrian-specific model, specific to the Pittsburgh area to see how each fares and which is best for our specific needs. To correct the issue with detection reflections as humans, being that having the bus in the image is not necessary for any of our analysis, we will simply only perform our detection in a section of each frame without the bus in it.

ACKNOWLEDGMENTS

This research is based upon work supported by the National Science Foundation under Grant No. 1659774. We thank the Robotics Institute Summer Scholars program for making this experience possible, Rachel Burcin and John Dolan for their support and guidance throughout the program and Albert Ye and Tom Bu the master students we worked under.

REFERENCES

[1] M. Paul, S. Haque, and S. Chakraborty, "Human detection in surveillance videos and its applications: a review," *Eurasip Journal on Applied Signal Processing*, vol. 176, pp. 1–16, 2013.

[2] I. Hasan, S. Liao, J. Li, S. U. Akram, and L. Shao, "Generalizable pedestrian detection: The elephant in the room," 2020.

[3] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, X. Zhao, and T.-K. Kim, "Multiple object tracking: A literature review," 2017.

[4] M. E. Hussein, W. Abd-Almageed, Y. Ran, and L. Davis, "Real-time human detection, tracking, and verification in uncontrolled camera motion environments," *Fourth IEEE International Conference on Computer Vision Systems (ICVS'06)*, pp. 41–41, 2006.

[5] Z.-Q. Zhao, P. Zheng, S. tao Xu, and X. Wu, "Object detection with deep learning: A review," 2019.

[6] C. Ye, "Busedge: Efficient live video analytics for transit buses via edge computing," Master's thesis, 2021.

[7] "Autonomous driving amp; adas (advanced driver assistance systems)." [Online]. Available: https://www.mobileye.com/

[8] J. Lee, J. Bang, and S.-I. Yang, "Object detection with sliding window in images including multiple similar objects," in *2017 International Conference on Information and Communication Technology Convergence (ICTC)*, 2017, pp. 803–806.

[9] M. Afifi, Y. Ali, K. Amer, M. Shaker, and M. Elhelw, "Robust real-time pedestrian detection on embedded devices," 2020.

[10] A. Quillen, "Roadbotics: Make data-driven decisions," Aug 2021. [Online]. Available: https://www.roadbotics.com/

[11] S. Zhang, Y. Xie, J. Wan, H. Xia, S. Z. Li, and G. Guo, "Widerperson: A diverse dataset for dense pedestrian detection in the wild," *CoRR*, vol. abs/1909.12118, 2019. [Online]. Available: http://arxiv.org/abs/1909.12118

[12] S. A. Velastin, R. Fernández, J. E. Espinosa, and A. Bay, "Detecting, tracking and counting people getting on/off a metropolitan train using a standard video camera," *Sensors*, vol. 20, no. 21, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/21/6251

[13] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2016.

[14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016.

[15] ——, "You only look once: Unified, real-time object detection," 2016.

[16] G. Tang, S. Liu, I. Fujino, C. Claramunt, Y. Wang, and S. Men, "H-YOLO: A Single-Shot Ship Detection Approach Based on Region of Interest Preselected Network," *Remote Sensing*, vol. 12, no. 24, p. 4192, 2020. [Online]. Available: https://hal.archives-ouvertes.fr/hal-03096535

[17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," *Lecture Notes in Computer Science*, p. 21–37, 2016.

[18] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," 2017.

[19] Z. Wang, L. Zheng, Y. Liu, Y. Li, and S. Wang, "Towards real-time multi-object tracking," 2020.

[20] Y. Zhang, C. Wang, X. Wang, W. Zeng, and W. Liu, "Fairmot: On the fairness of detection and re-identification in multiple object tracking," 2020.

[21] K. Christensen, "Computer vision for live map updates," Master's thesis, 2019.

[22] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: A benchmark," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 304–311.

[23] S. Zhang, R. Benenson, and B. Schiele, "Citypersons: A diverse dataset for pedestrian detection," 2017.

[24] H. Qu, M. Wang, C. Zhang, and Y. Wei, "A study on faster r-cnn-based subway pedestrian detection with ace enhancement," *Algorithms*, vol. 11, p. 192, 11 2018.

[25] S. Velastin, R. Fernández, J. Espinosa, and A. Bay, "Detecting, tracking and counting people getting on/off a metropolitan train using a standard video camera," *Sensors (Basel, Switzerland)*, vol. 20, 2020.

[26] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016.

[27] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," 2019.

[28] E. M. Silva Machado, I. Carrillo, M. Collado, and L. Chen, "Visual attention-based object detection in cluttered environments," in *2019 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, 2019, pp. 133–139.

[29] S. Zhang, Y. Wu, C. Men, N. Ren, and X. Li, "Channel compression optimization oriented bus passenger object detection," *Mathematical Problems in Engineering*, vol. 2020, pp. 1–11, 03 2020.

[30] Y. Wang, K. Kitani, and X. Weng, "Joint object detection and multi-object tracking with graph neural networks," 2021.