Centroid-Tracking-Aided Robust Object Detection for Hospital Objects

Fabiola Maria Teresa Retno Kinasih
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
fabiola.maria@students.itb.ac.id

Carmadi Machbub
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
carmadi@lskk.ee.itb.ac.id

Lenni Yulianti
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
lenni@lskk.ee.itb.ac.id

Arief Syaichu Rohman
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
arief@lskk.ee.itb.ac.id

2020 6th International Conference on Interactive Digital Media (ICIDM) | 978-1-7281-4928-8/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICIDM51048.2020.9339679

***Abstract*—COVID-19 outbreak has a big impact to people's daily life in 2020, especially in healthcare sector. As COVID-19 viruses are highly contagious, it is important to take strict measures to ensure all patients got the needed care while taking healthcare workers safety into consideration. Robot-based care is being hurriedly developed recently, and one of the important abilities for such robot is to be able to distinguish object commonly found in hospital thus the robot can make the correct action towards the correct object. For this publication, an object detector is trained to detect the hospital bed, thus it can be an input to the care robot navigation system when it is going to approach patients. As hospital beds vary from one brand to another, and this research has limited time constraint and readily available hardware, the object detector confidence is still low. Thus, a centroid tracking method is implemented to aid the hospital object detection, ensuring the robot can detect the correct bed more robustly with considerable speed for embedded implementation.**

Keywords— movement distance score, object detection, object tracking, speed (in frame per second)

I. INTRODUCTION

Recent development on the CNN based object detection allows student and researcher to train custom object detector crafted to suit their application needs with considerable accuracy and less time due to the transfer learning concept. Here the pre-trained weight can be furtherly trained to acknowledge new objects such as hospital bed or other objects with shorter training time, as only values on several last layer needs to be adjusted to accommodate new object. As the goal is similar, to distinguish object captured by visible light camera, the base layer from another object detection can still be used as the base layer of the new object detector.

This paper is a part of Object Tracking research in Control System and Computer Laboratory, ITB. Previous works has been done in face detection and tracking[1], [2], custom object tracker[3], a combination of face and posture to track human[4], and a state-machine based image processing method combination[5]. The goal of this current research project is to detect hospital beds and give them identity thus robot can distinguish those beds. Regarding the goal of this project, it is beneficial to exploit the ability to classify the objects from object detector, thus automatically choose all area containing hospital bed as the target object, while maintaining the ability to distinct between one object and another, even if those objects belong to the same class.

The object detector being used is the open-sourced tensorflow implementation of You Only Look Once: Darkflow[6]. Darkflow is choosen instead of Darknet as tensorflow is rather user friendly for researcher compared to darknet while maintaining the compatibility with GPU and comparable accuracy. The architecture of YOLO will be explained in section 2: Object Detector. Transfer learning concept that is being used in this paper will be briefly discussed in the end of section 2. Centroid tracking that is being used for its lightness will be discussed in section 3. The implemented system and its result will be explained in section 4, followed subsequently with conclusion.

II. OBJECT DETECTION

Before the publication of AlexNet [7] in 2012, the commonly used method are Haar-Cascade based Viola Jones[8] that was published in 2001, Scale-invariant feature transform [9] (SIFT) that was published in 1999, and Histogram of Oriented Gradients for pedestrian detection [10] in 2005.

TABLE I. ACCURACY COMPARATION (FOR NON DEEP LEARNING)

Method	Accuracy	Speed
Viola-Jones	74.8% (for face) [11]	33 fps (our trial using GPU), 2 fps (handheld devices on original paper[8]), 15 fps (CPU Pentium III for 288*384 pixel)
HOG-SVM	80% (our trial)	16 fps on GPU, less than 10 fps on CPU

Up until now, the feature extractor from those three methods are often used in various application in computer vision field. But the trend for learning method to distinguish one object and another has been sifted, from Adaboost and SVM into deep learning trend. The first CNN based method for GPU implementation was published in 2004[12], which was 4 times faster than its equivalent CPU implementation. However, the You Only Look Once [13] (YOLO) method publication in 2014 makes the CNN based object detection more popular and commonly discussed even among researcher in less developed nations. As GPU price has become more affordable over the years, the implementation of

CNN based object detector are becoming more feasible. Take into consideration that recent CNN-based detector has higher accuracy and performance, choosing CNN-based architecture for object detector is a rational option.

TABLE II. ACCURACY COMPARATION (FOR DEEP LEARNING)

Method	Accuracy	Speed
AlexNet	84.7% on ILSVRC 2012	700 fps (on GPU Nvidia Titan), 34 fps (on GPU Nvidia Jetson) [14]
Mobilenet v2	71% (top 1) 91% (top5) on ILSVRC 2012	299 fps (on GPU Nvidia Titan), 48 fps (on GPU Nvidia Jetson) [14]
YOLO	76% (top 1) 92% (top5) on Pascal VOC 2007	45 fps (on GPU for 448x448 pixels sized images) [13]
SSD	87.9%	46 fps (on GPU for 300x300 pixels sized images) [15]

Considering the accuracy, and possible accuracy reduction while doing transfer learning, YOLO architecture is chosen for this project. Tensorflow implementation of YOLO (Darkflow) is choosen as tensorflow provides a robust framework with various choice of CNN architecture for transfer learning compared to Darknet.

A. You Only Look Once

You Only Look Once (YOLO) architecture is being choosen as it has highest accuracy and is proved successfully handle task to classify over 9000 kinds of object. It combines Fully Connected Feedforward Network and recently popular Convolutional Neural Network combined with MaxPooling method to reduce computational cost.

The basic components of YOLO are:

- Image resizing (thus all image inputs will be same-sized)
- Convolutional layer (as feature extractor)
- Max pooling (to prioritize important/dominant features)
- Fully Connected layer (to classify)

Besides classifying objects, an object detector should predict how likely their prediction are correct. This usually referred to as "confidence" scores. YOLO considers Intersection of Union as a variable to determine the confidence score, formularized as follows:

$$confidence = \Pr(Object) * IOU_{pred}^{truth} \quad (1)$$

In the testing process, the conditional class probabilities is multiplied with the individual box confidence predictions, thus the confidence scores are specific for each class:

$$\Pr(Class_i | Object) * \Pr(Object) * IOU_{pred}^{truth} =$$

$$\Pr(Class_i) * IOU_{pred}^{truth} \quad (2)$$

Here is the example of YOLO detection bounding box and confidence scores. The object class will be displayed, along with the confidence scores that this object is really belong to the class predicted.

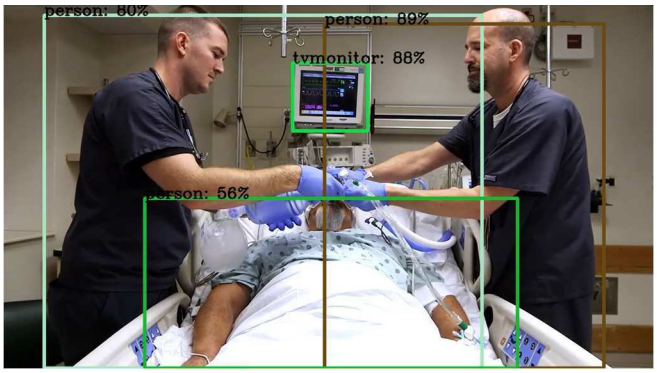

Fig. 1 Example of YOLO (darkflow implementation) performance.

Below is the architecture of YOLO

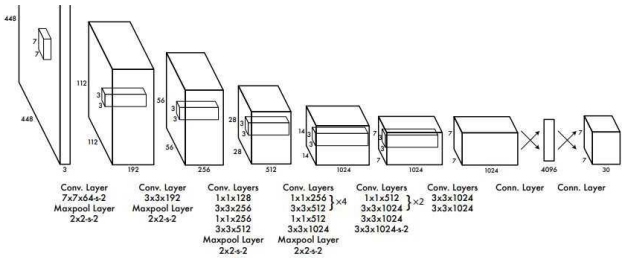

Fig. 2 YOLO CNN architecture.

For the neural network YOLO employs linear activation function for the final layer, and a modified rectified linear activation function for all preceeding layers as formularized below:

$$\Phi(x) = \begin{cases} x, if\ x > 0 \\ 0.1x, otherwise \end{cases} \quad (3)$$

The limitations of YOLO exists due to each grid cell (small area of the input image) can only predicts two boxes and can only have one class. Thus, YOLO is not quite effective when used to detect many small objects reside in a small area.

B. Transfer Learning

In machine learning, a trained model sometimes being used in training a model for different but related task. The most common example is the object detection, pretrained weight for detecting object sometimes being used to classify X-Ray images, CT-Scan images, detecting different kinds of animal, or in this research project, to detect common things find in a hospital. A previously acquired knowledge in differentiating object from the established object detector is transferred to a new model intended to detect hospital/health-related object specifically. Note that this only works for related task with similar goals [16]. Transfer learning used in deep learning (in this case, YOLO architecture) is called inductive transfer. Model bias is narrowed (intentionally) using a model fit on a different but related task (still detecting object, but healthcare related instead of daily-life objects).

This transfer learning method might works as both tasks (classifying daily-life objects and healthcare-related objects) are utilizing the similar features, both family might differentiate each object class belong to them by its edges, corners, intensity difference, or other indescribable features from human point of view that only makes sense by the way a computer program works. Thus the weight for the baselayer (the earlier layer) carry some meaning in the new model trained for the new specific task. However, as the final goal,

in this case the object to be classified are different, the last layer should be tuned to suit the new object class belong to the new task. Below is the possible performance increase (gentler learning curve) due to transfer learning.

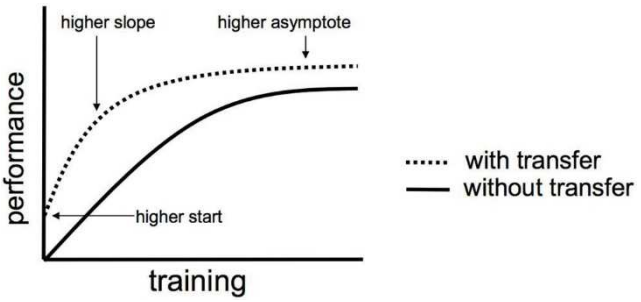

Fig. 3 Performance improvement with transfer learning.

III. Tracking Method

A. Tracking versus Object Detection

As tracking is intended for associating target object that appears on sequential video frames with ID preserved, it is different with object tracking with goal to classify object that appears in a frame to each corresponding category. Thus, tracking method is commonly faster as it has knowledge about previous object location to determine the next probable location, is better at hanndling occlusion due to its predictive nature, and can preserves identity by harnessing location information.

TABLE III. Tracking versus Object Detection

Tracking	Detection
Commonly Faster (10-100 fps for correlative filter tracker, and can be higher for color, centroid, or predictive filter tracker such as Kalman Filter) Reason: Has acknowledge specific object location from previous frames	**Relatively Slower** (10-45 fps for deep learning based method, 50 fps or more for Viola Jones running on modern machine) Reason: Has to re-detect all object for each subsequent frames (from a broad class model)
Better at handling occlusion	The bounding box will immediately vanish if the object appearance did not match the (broad) class model in the database
Preserving identity (ID 1 Object and ID 2 Object will be consistent across frames)	There is no identity preservation, re-detection is needed for each new frame

Below is the commonly implemented tracking method pipeline

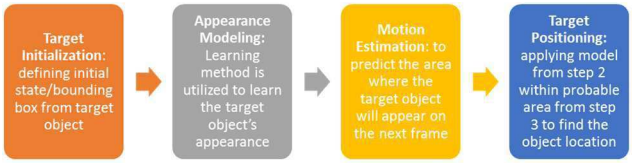

Fig. 4 Common recent tracking method pipeline.

However, for old method such as centroid tracker, the second steps is neglected, as the motion estimation is the main task for this algorithm.

B. Centroid Tracking

Although many recent methods are evolving in the Tracking method, both in Machine Learning family such as MDNet, Rolo, etc and Correlative Filter Tracker such as MOSSE, KCF, and DCF-CSR with higher accuracy and ability to be run stand alone, Centroid tracking is chosen in this project due to its lightness. As the target object in this research project are plenty (there are numerous healthcare object in a hospital room) and stands still, with moving camera (due to robots move) it is best to keep the object detection run for every frame. Thus, the old centroid tracking method is still applicable, and has advantages for its impeccable speed (up to hundreds of fps on today's machines even without GPU)[17].

The centroid tracking method will accept the bounding box from the object detector, calculating the centroid. To assign the ID, previous centroid location are being used. The nearest previous centroid location will be assigned the same ID as before (as long as they belong to the same class). For new centroid location, commonly caused by a new object that was not visible/detected in the previous frame, a new ID will be assigned. This implies that for the first frame, new ID will be assigned for all detected objects. As the robot moves and the camera views shifted, some object that was detected will disappeared or no longer got detected. The ID for disappeared object will be de-registered, so when the same object got visible again later, it might get assigned different (new) ID. This is the most workable solutions considering there are plenty of the exact same object being used for different patients such as bed, syringe, infusion stands, desks, curtain, etc.

IV. Implementing the Methods Combination

As studied in my previous publication, running object detector and tracker for each frames is quite computation extensive compared to the simple state machine approach or if the object detection only implemented for the first frame. For each frame retrieved, the frame will be processed through the object detection and tracking process, as shown follows.

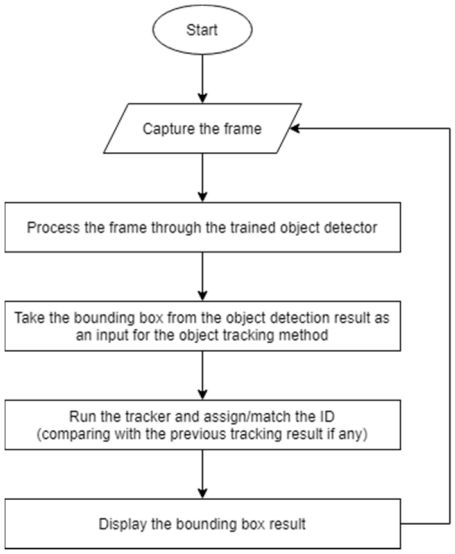

Fig. 5 Flow diagram for the program being used in this project.

Trading off the speed, this method offers better accuracy than simple state machine configuration as the object detection (classification) is done for each frame. The ID will be assigned or re-matched after re-detection process, instead

of only considering the previous tracking result as in the simple state machine configuration when the program is in the "tracker" state. However, as the lightest tracker is chosen in this research project, the speed is mainly depends on the speeds of YOLO implementation on that machine. For fully embedded implementation, tiny YOLO might be better option for the system to run smoothly.

V. Results and Discussion

A. Trained Object Detector Result

After some training with 200 images of empty hospital beds, the YOLO-based object detector is able to detect the hospital beds, even with patients lay on it and other objects are present, with quite good performance but low confidence scores.

Fig. 6 Object detector detecting hospital bed from left-side.

Fig. 7 Object detector detecting hospital bed from the front.

Fig. 8 Object detector detecting hospital bed from the right-side.

The low confidence scores are one of the main reason why it would be a bit risky to not running the object detector for every frame, thus we decided to run both object detector and object tracker for every frame.

B. Lightweight Centroid Tracking Result

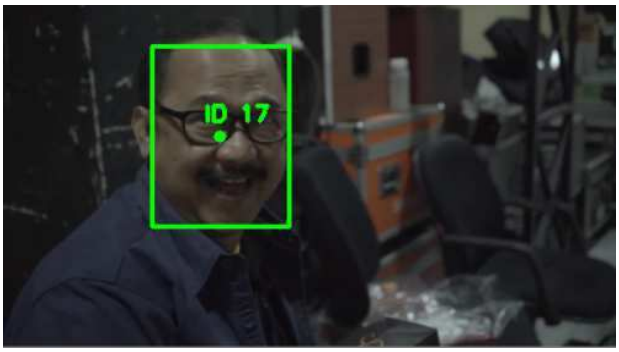

Fig. 9 Example of centroid tracking performance.

This is the example of centroid tracking performance while accompanied with other kinds of CNN based object tracker with SSD architecture used to detect face. Although the objects tracked belong to the same class, specific identifier can be given to each individual object.

C. Centroid Tracking Assisted Object Detector

Although further study is still needed, it is interesting to add additional meter besides confidence scores. This new score comes from considering movement distance score, and formularized as follows:

$$movement\ distance\ score = \frac{|current\ horizontal\ centroid - previous\ horizontal\ centroid|}{detected\ object\ width} \quad (4)$$

Fundamentally, this score calculates displacement compared to the object size. This is due to the nature of nearer object that will looks bigger, and small displacement will cost much pixel, compared to farther object that will looks smaller, and small pixel count displacement might already means significant displacement in real life. Act as a normalization coefficient, this help given us better real-life displacement score, with smaller score means small or no displacement, thus more likely for the object to be the same object.

Fig. 10 First frame of movement distance frame calculation.

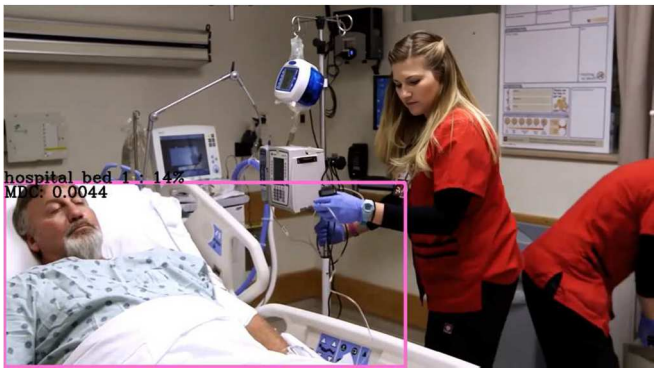

Fig. 11 Second frame of movement distance frame calculation, same point of view.

The first frame has big movement distance score as there is no previous centroid tracked, thus the movement distance score will be the default value 0.5. Subsequent score is significantly lower as the bed is stationary, confirming that the system is still tracking the same object as before.

Fig. 12 Subsequent frame of movement distance frame calculation, slightly different point of view causing horizontal displacement.

Slight point of view changes will increase the movement distance score, but is still considerably small. Thus, we are affirmed that the system is still tracking the same object as before.

However, this score is calculated with stationary target object tracking in mind. Hence this score is not suitable for tracking fast-moving object as the location will significantly differ in such short period. Motion estimation will be a better assist in those cases. Also, it is worth noting that width is chosen instead of height as the denominator, as the robot movement will is 2-dimensional, thus significant horizontal displacement is more likely to happens rather than significant vertical displacement.

VI. Conclusion

Although the movement distance score does not directly alter the confidence scores, it can help user or other developer to get better idea on how good the custom object detector works for tracking stationary object with low-speed vehicle. It is also worth noted that centroid calculation is very lightweight compared to the object detector inference time thus it does not add significant process time.

Acknowledgment

The authors thanked the Centre of Research and Community Service of the Institut Teknologi Bandung (LPPM ITB) for funding this research through the *Program Penelitian dan Pengabdian Masyarakat* (Research and Community Service Program) scheme.

References

[1] S. R. Yosafat, C. Machbub, and E. M. I. Hidayat, “Design and implementation of Pan-Tilt control for face tracking,” 7th IEEE International Conference on System Engineering and Technology (ICSET), 2017.

[2] D. A. Maharani, C. MacHbub, and P. H. Rusmin, “Enhancement of missing face prediction algorithm with kalman filter and DCF-CSR,” 2019.

[3] S. Suryadarma, T. Adiono, C. Machbub, and T. L. R. Mengko, “Camera object tracking system,” 1997.

[4] D. Andriana, A. S. Prihatmanto, E. M. I. Hidayat, and C. Machbub, “Combination of face and posture features for tracking of moving human visual characteristics,” Int. J. Electr. Eng. Informatics, 2017.

[5] F. M. T. R. Kinasih, C. F. D. Saragih, C. Machbub, P. H. Rusmin, L. Yulianti, and D. Andriana, “State machine implementation for human object tracking using combination of mobilenet, KCF tracker, and HOG features,” Int. J. Electr. Eng. Informatics, 2019.

[6] T. H. Trieu, “Darkflow,” GitHub Repos, 2018. [Online]. Available: https//github.com/thtrieu/darkflow. [Accessed: 14 Febr. 2019].

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “2012 AlexNet,” Adv. Neural Inf. Process. Syst., 2012.

[8] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” 2001.

[9] D. G. Lowe, “Object recognition from local scale-invariant features,” 1999.

[10] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” 2005.

[11] A. M. Basbrain, J. Q. Gan, and A. Clark, “Accuracy enhancement of the viola-jones algorithm for thermal face detection,” Lecture Notes in Computer Science, pp. 71–82, 2017.

[12] K. S. Oh and K. Jung, “GPU implementation of neural networks,” Pattern Recognit, pp. 1311-1314, 2004.

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

[14] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, “Benchmark analysis of representative deep neural network architectures,” IEEE Access, 2018.

[15] W. Liu et al., “SSD: Single shot multibox detector,” Lecture Notes in Computer Science, pp. 21–37, 2016.

[16] S. J. Pan, “Transfer learning,” in Data Classification: Algorithms and Applications, 2014.

[17] A. Rosebrock, “Practical Python and OpenCV Case Studies,” Pyimagesearch.Com, 2015.