Angular Position Estimation for Human-Following and Robot Navigation

Isaac Asante(✉), Lau Bee Theng, and Mark Tee Kit Tsun

Faculty of Engineering, Computing and Science, Swinburne University of Technology, Kuching, Sarawak, Malaysia
iasante@swinburne.edu.my

Abstract. Mobile robot navigation and human following are two related areas under the field of robotics that have garnered a lot of interest over the years, due to their advantages in the real world, in various settings. Modern techniques depending exclusively on computer vision for environmental data input commonly rely on state-of-the-art object detection methods to obtain data about the location of detected objects in a scene. However, these detection frameworks do not directly provide the angular position of obstacles and human targets in images. In this research project, the Mask R-CNN instance segmentation framework detects static objects and humans in an environment. A chain of algorithms is then used to transform the image's content and pixel information into a one-dimensional array that can be mapped to the robot's field of view. The findings show that the result can aid a mobile robot in estimating the angular position of obstacles and a human in a scene, which is necessary for collision-free navigation and robot-human following in an unknown environment. The proposed method also shows adaptivity as it works outdoors and indoors under poor lighting conditions. It can be used as a standalone algorithm in robotics simulations with webcams and static images.

Keywords: Robot Navigation · Instance Segmentation · Object Detection

1 Introduction

The research topics associated with developing an autonomous companion robot can be subcategorized for a direct appreciation of the dilemmas that pervade the subdomains of this area of interest. Locomotion, perception, path planning [1], and continuous target tracking [2] all constitute the fundamentals of robot-human following activities, regardless of the nature of the environment. This research focuses on perception, which consists of obtaining environmental data to aid the robot in building knowledge of its surroundings. This perception involves the robot sensing entities around itself – be it a human target or objects posing as dynamic or static obstacles – along with their position relative to itself [3].

D.-J. Deng et al. (Eds.): SGIoT 2022, LNICST 497, pp. 279–290, 2023.
https://doi.org/10.1007/978-3-031-31275-5_26

2 Background

Cameras have been utilized for many years in mobile robot navigation projects to receive visual input and build algorithms to help a wheeled robot navigate unguided while avoiding collisions indoors or outdoors. Monocular and stereo cameras have served well for such objectives [4, 5], and depth cameras have also been a common choice for modern techniques [6–11]. Knowing who to track and where to navigate to avoid collisions is indispensable for a human-following mobile robot. This means that the distinction between a human and an object is vital. This implies that object detection is at the core of the activity, and state-of-the-art machine learning models, such as those made available via the TensorFlow Object Detection API [12], offer viable solutions for this very challenge.

Nonetheless, robust open-source detection models such as CenterNet [13], EfficientDet [14], SSD MobileNet [15], SSD ResNet [16], and Faster R-CNN [17] typically output bounding boxes [18], for which the coordinates normally provide information as to where the classified objects are detected in an image. Thus, in an uncontrolled environment where objects may not necessarily be shaped as cubes or cuboids, it becomes theoretically unreliable for a robot to base its estimation of an obstacle's angular position relative to its camera solely on the object's bounding box upon detection by a trained model. Image segmentation appears to be practical in addressing this shortcoming. In recent years, it has become common to employ segmentation techniques as part of systems developed for person or object tracking. Facebook AI Research's Mask R-CNN framework has been a recurrent choice in such studies. It extends from Faster R-CNN and can generate instance segmentation masks at considerable speeds running at five frames per second and achieving mask AP of 35.7 when tested on the COCO dataset with a ResNet-101 backbone [19]. This makes it efficient for general video segmentation and real-time segmentation on a live camera feed, such as in the case of a navigating mobile robot equipped with a visual sensor. In [20], Mask R-CNN is used for classification on Video Instance Segmentation (VIS), along with an adaptation of the network by [21]. The framework can further be extended to improve its various components, as shown in [22], where an enhanced information propagation method is built on top of Mask R-CNN.

The diversity of robust detectors and low-power sensors today suggests that end-to-end lightweight robot navigation and human tracking systems can be developed using state-of-the-art frameworks and techniques, and existing ones can be improved. With cameras being mainstream in robotics as input sensors, it is worth exploring new ways of pairing instance segmentation methods with computer vision algorithms to achieve high-performance object classification, human tracking, and collision-free navigation.

Table 1 presents the results of evaluations of the proposed angular position estimation algorithm for different RGBD cameras commonly used in the robotics industry [23]. The results indicate that at lower resolutions on a machine equipped with a Tesla T4 graphics processing unit, the algorithm can match Mask R-CNN's execution speed of 5 fps [19].

Table 1. Angular position estimation algorithm evaluation

Camera	RGB Resolution	FOV	Total New Feature Maps	1D Matrix Data Points	Execution Time (In Milliseconds)
Microsoft® Kinect™ 2.0	1920x1080	70°H, 60°V	3	60	842
ASUS® XtionPro™ Live	640x480	58°H, 45°V	3	80	171
Intel® RealSense™ Camera D455	1280x800	90°H, 65°V	3	80	439
Orbbec® Astra Mini™	640x480	60°H, 49.5°V	3	80	185
roboception® rc_visard™	1280x960	61°H, 48°V	3	80	517
MYNT EYE	752x480	146°D, 122°H, 76°V	3	94	201

3 Proposed Solution

This paper presents a fast way of processing two-dimensional image data from a camera to estimate the angular positions of objects and a target human in a scene relative to a robot's camera. The proposed method is designed in a simulated environment, which therefore eliminates the need for a real human subject, or additional elements such as active markers [24]. The results connote a strong relationship between the contents of an input image obtainable through instance segmentation masks, the image's original shape, and the camera's field of view (FOV). Computing the data from this triad of information using this paper's chain of algorithms makes it possible to estimate the angular position of elements in a scene relative to the mobile robot's central field of vision.

The proposed solution uses the TensorFlow backend with Keras and Python 3 and employs the Mask R-CNN framework. The latter is suitable for the project's objective, as the required model output for the expected result is a set of instance segmentation masks corresponding to regions containing persons or static objects in the input image. Here, the input images are assumed to be frames captured by a camera mounted on a mobile robot at specific intervals. The goal of this phase is not to implement an end-to-end mobile robot navigation system but to demonstrate how two-dimensional information from an image is enough to control a robot's rotation angle as it navigates without substantial computational overhead. As the eventual human-following system is set to be fully implemented in the Robot Operating System (ROS), this can be achieved by accurately estimating the yaw value that should be published to the mobile robot's odometry topic subscribed. Though odometry messages representing a robot's orientation in free space generally use quaternions [25], it is possible to obtain Euler angles from quaternions in

ROS using the Transformations library [26]. During the image processing part of this project, inferences are run on one image at a time per GPU. With a single GPU, the batch size is set to 1. A pre-trained model is used instead because training an object detection or segmentation model on a standard collection such as the Microsoft COCO dataset can be time-consuming [27]. For efficacy, the solution relies on the pre-trained weights from the Mask R-CNN 2.1 release [28] and functions from the source code files available on the GitHub repository [29]. The backbone utilized is ResNet-101, which has been measured with higher mean average precision than other renowned convnet architectures such as VGG-16 [30] in benchmark tests involving both the PASCAL VOC 2007 and 2012 and COCO datasets. The class names for the detections list include labels of several objects commonly found in indoor spaces. Ensuring that labels for indoor items such as furniture are included among the set of classes recognizable by the model is vital. This is because the proposed solution explained is meant to be part of an indoor robot navigation model in subsequent phases of this research.

3.1 Merging Segmentation Masks

The predictions from the model contain the segmentation masks for every classification in its matrix, grouped in a container of shape *(h, w, n)*, where *h* is the height of the original input image, *w* is the image's width, and *n* is the total number of masks returned by the prediction. Visualizing the results clarifies how the various instances can be depicted in a single two-dimensional matrix.

The solution's objective is to yield knowledge about a scene during robot-human following—specifically obtaining knowledge about the presence of a human and obstacles in a scene. Therefore, the segmentation masks must be consolidated. Because the masks' matrices initially consist of boolean values, converting them to integers restricts them to two equivalent possibilities: 1 or 0. Any value in the converted matrix that reads 0 points to an area in the original image where the pixel did not belong to the mask class. Conversely, a value of 1 denotes a pixel belonging to the mask in question—or put differently, it indicates that it corresponds to a recognizable class in that region of the original image. However, this is not enough to distinguish between a person and an obstacle during robot navigation. The next step to address this drawback is to re-assign the numerical values for each mask based on their class, per the class names from the COCO dataset used to train the Mask R-CNN model.

Where the value is currently one after the switch in data type from boolean values to integers, a higher positive value is assigned if the name of the class associated with the mask is 'person'. Otherwise, a large negative value is assigned because the mask refers to a static object or obstacle the mobile robot must avoid during navigation. The value is left intact for other areas where the pixel value is 0. This methodology uses default values $\{-4, 0, 4\}$ to update each mask from the prediction results per the mentioned conditions (Table 2). Upon updating the integers for all masks, any matrix corresponding to an obstacle's mask is expected to comprise only the unique values $\{-4, 0\}$. On the other hand, a human's mask would have the unique values $\{0, 4\}$. This pixel re-assignment step is indispensable to the pooling operation illustrated in Fig. 1.

The subsequent step consists of matrix additions to consolidate all instance segmentation masks into one 2D matrix. The latter aims to hold pixel information about the

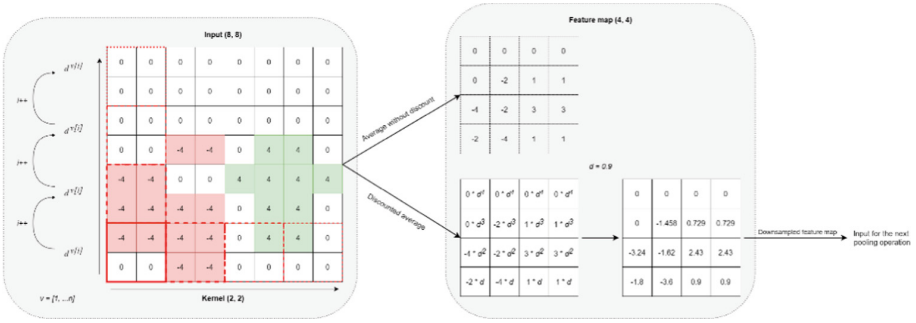

Fig. 1. Illustration for the average pooling algorithm. The discount factor d gets raised to a new power (incremented by 1) at every vertical stride. This sample uses an input matrix of shape (8, 8) and a kernel of shape (2, 2) to generate a down-sampled featured map of discounted averages of shape (4, 4). In practice, input images would be much larger, and dynamic kernel selection would be required to optimize the downsampling process.

three derived groups of object detections of interest for collision-free robot navigation: Obstacle, Void, and Human. This information was retained in the previous step through the pixel values $\{-4, 0, 4\}$ used in the re-assignment function. In a project with different conditions, the length and values in the set of pixels would have to be different to reflect the main objectives.

Here, the sum of all masks results in a new matrix of shape *(h, w)*, where *h* is the height of the original input image, and *w* is the original width, just as before. For clarification, all masks in the prediction results had the same shape. It is not uncommon for generated masks to have overlapping pixels [20], and although this may be an inconsequential phenomenon in some ways, it is to be anticipated. Whenever this happens, the sum of matrix elements may produce values lower than -4 or higher than 4. For example, despite each mask having either $\{-4, 0\}$ or $\{0, 4\}$ as a set of unique values, as explained before, the resulting matrix may be composed of the set $\{-8, -4, 0, 4, 8\}$, due to overlapping pixels.

Table 2. Assigning pixel values in masks

New Values	Purpose
-4	A pixel belongs to a static object or obstacle in the current mask's matrix
0	A pixel that does not belong to any detection in the current mask's matrix. It may indicate free space, belong to other masks in the prediction results, or refer to objects not detectable by the Mask R-CNN model in the original image
4	Pixels that belong to a human detection in the current mask's matrix

3.2 Pooling Operations

The extraction of valuable data to estimate the angles of entities in a scene relative to the mobile robot is performed through an indefinite number of pooling steps. The latter relies on a set of recommended sliding windows holding no weights. This is analogous to kernels having all their weights set to 0. They are not used as filters to extract features from the matrix sum as it would be done with a traditional CNN approach [31], but they still produce down-sampled feature maps through average pooling. In this case, the shapes of the proposed default kernels in order of the largest to the smallest are (8, 8), (4, 4), and (2, 2).

The strides required to slide across an input matrix horizontally and vertically are calculated using integer divisions. The input's height is divided by the kernel's height, whereas the input's width is divided by the kernel's width, as illustrated in Fig. 1 above. Nevertheless, to adapt the algorithm to the problem at hand, collision-free robot navigation and human following, the pooling operations start on the input's bottom left side. The vertically-reversed pooling operation is to get a discounted average for the fitting patches of the input feature map. The discount factor introduced here gradually lessens the pixel averages in different map regions. At every stride in an upward direction, the discount is amplified. Theoretically, this pooling technique lets objects closer to the mobile robot's base take precedence over those farther away or positioned higher off the ground. The rationale behind this averaging method is that, as a wheeled agent without the ability to fly, the mobile robot is bound to maintain ground contact at all times. Obstacles on the floor at a near distance pose a much higher risk of collision than those that are either further away or high enough for the robot to pass underneath them.

Figure 1 presents a sample matrix sum conveying obstacles represented as red cells with negative values; a human represented as green cells with positive values; and free space as white cells with zeros. The pooling operation starts from the matrix's bottom-left corner using a kernel of shape (2, 2) and moves towards the right before making an upward stride to calculate the averages from left to the right. This step is repeated until the sliding window reaches the input matrix's top right corner. As the matrix is of shape (8, 8), the kernel can slide four times to the right and four times upward without any overlap for a total of 16 average values stored correspondingly in a smaller shape matrix (4, 4). In this illustration, the array v denotes the sequence of vertical strides possible, from 1 to n, starting from the bottom. Each value in the sequence is referenced as $v[i]$ in Fig. 1, with i representing each value's index inside the array v. The discount factor's default value is initialized to 0.9. Any time the kernel makes a stride up, the discount factor is raised to the power of $v[i]$. In this case, the discount value goes from 0.9^1 to 0.9^4; that is, from 0.9 to 0.6561. Therefore, all the average values calculated by the sliding window in the bottom row are multiplied by a discount factor of 0.9.

In contrast, the values in the topmost sliding row are multiplied by a discount factor of 0.6561. In practice, this algorithm takes a feature map. It provides a down-sampled version of it to use as input for further processing, as visualized in Fig. 2. Nonetheless, performance must be considered, and it must be noted that excessive down-sampling may occur in two ways, as explained below.

Firstly, relying on the larger default kernel of shape (8, 8) alone may produce smaller feature maps rapidly, but this is bound to cause significant information loss. Secondly,

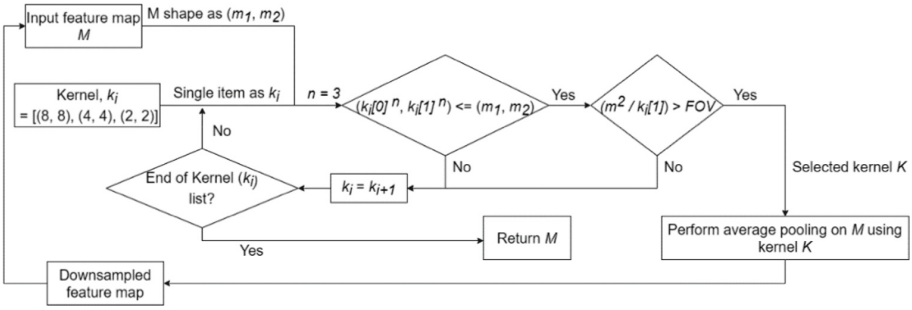

Fig. 2. A breakdown of the kernel selection algorithm, which ensures that the best kernel is used at every step to recursively perform the discounted average pooling operation. The processing stops once the feature map gets smaller than the field of view of the robot's camera. Here, the exponent n in the first condition is recommended to be set to 3.

relying solely on the smaller kernel of shape (2, 2) may retain a lot of useful information but may require too many iterations, thus becoming impractical. Consequently, the proposed technique introduces a kernel selection algorithm for each iteration. As explained in Fig. 2, the aim is to maintain a balance between rapid downsampling and information retention at every step. Once no more kernel can be fitted into the returned feature map, the column-wise sum is calculated and used to determine which part of the captured scene must be avoided from left to right. The logic is that the final 1D matrix generated by the column-wise addition represents the robot's rotation range r in degrees, as noted in (1). The middle of the range (0°) represents the robot's central field of vision.

$$r = [-1 * (FOV/2), FOV/2] \tag{1}$$

Nevertheless, to further speed up the downsampling process and retain more information, a coefficient n is introduced, and the robot's camera's field of view FOV is added to the kernel selection algorithm. The recommendation for this method which yielded the most desirable performance results in early tests was $n = 3$, and the field of view of the camera used in this case was 58 degrees. In the final step, the 1D matrix is normalized and plotted in a graph for visualization purposes. As seen in Fig. 3, the high data points in the graph indicate areas from the original image that contain the strongest human presence, whereas the low points imply the presence of obstacles to avoid. The lower the point, the likelier it is to refer to an object close to the robot or an obstacle perceived as relatively large. In an indoor environment with only one human target and no other person, the single highest data point would indicate their angular position in the scene relative to the robot's camera. This can be seen in Fig. 3, where the human in the dark office room is detected by the model and assigned high data points in the graph (constituting a peak), as opposed to the detected office chairs causing dips in the plotted line. The aim is to map the length of the 1D matrix—its count of elements or the maximum value on the plot's X-axis—to the field of view of the robot's camera. This way, the middle coordinate on the X-axis becomes the 0-degree angle in the real world or the default rotation angle when the robot is fixated on a direction and not rotating. The actual 0 coordinate on the X-axis thus represents the far edge of what the robot

perceives on its left without rotating. The same applies to the maximum value on the X-axis and the right edge of the robot's captured image frames. This mapping logic is demonstrated in Fig. 4, in which the algorithm selects an angle of 21.375° upon mapping the length of the resulting 1D matrix (80 data points) to the camera's field of view (57°). In contrast with Fig. 3, Fig. 4 showcases the angular positioning algorithm in an outdoor environment, proving that the proposed method is suitable for indoor and outdoor settings. Indeed, the exact rotation angle can be calculated from the result's 1D matrix in degrees, and the equivalent can also be obtained in radians. In (2) and (3), x is the index of the highest value (or data point) in the final 1D matrix, and r is the range of the 1D matrix or the number of data points describing the scenes as seen in the fifth column of Table 1. FOV is the camera's field of view. Below is the mapping formula to obtain the angle of rotation in degrees:

$$deg = (x * (FOV/2))/(r/2) - FOV/2 \tag{2}$$

To obtain the equivalent value in radians, the value of π is introduced. This is useful, considering angles are typically computed in radians in ROS.

$$rad = ((x * (FOV/2))/(r/2)) * (\pi/180) - (FOV/2) * (\pi/180) \tag{3}$$

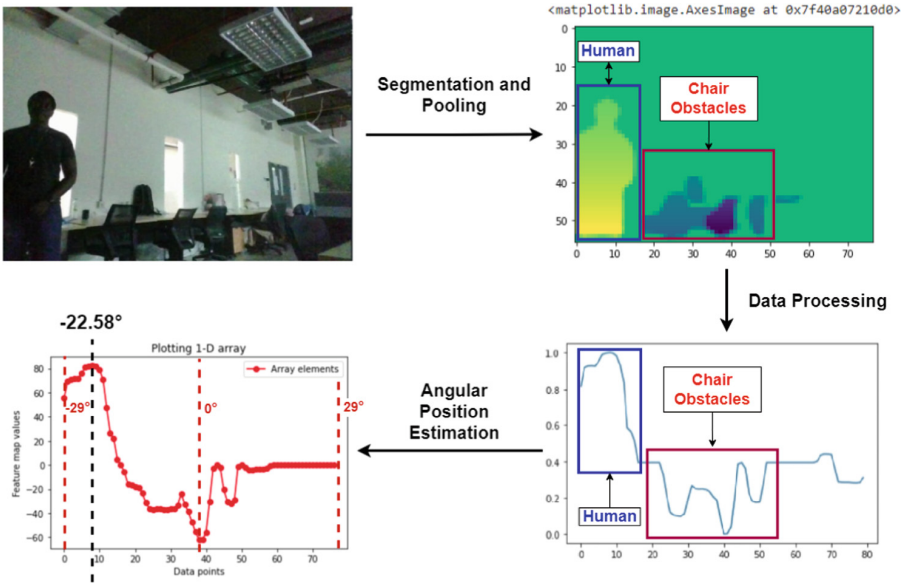

Fig. 3. Live snapshot from a test carried out in an indoor office under poor lighting conditions. The high data points are assigned to the human target. In contrast, the three detected chairs force three noticeable dips in the graph, providing the robot with information about areas to avoid. Here, the angular position estimation algorithm recommends that the robot rotate 22.58° to the left to get the human target in its central field of vision.

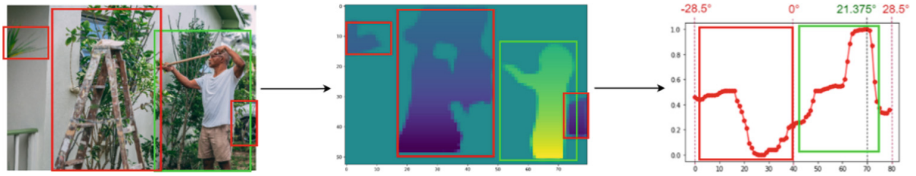

Fig. 4. Illustration of the angular position estimation algorithm on a static image portraying an outdoor environment. The segmentation data gets processed, and the results are fed to the mapping algorithm, which outputs the most suitable degree angle the robot has to turn to face the target. Here, the output is 21.375° to the right.

4 Conclusion and Future Work

The significance of this research work can be linked to the lack of vision-based scene perception techniques that can be employed in different settings, especially indoor robot navigation. Self-driving cars and assistive technology for the visually impaired are two notable domains that share many requirements that an autonomous wheeled robot would need to track a human target without collision. The relevance has to do with the sightless nature of the main agents involved, despite the scenes being prone to various elements to identify, avoid, and track [32–39]. Restricting a robot's perception strictly to a specific set of recognizable items limits the efficacy of the robot's model in the real world, as the robot is bound to encounter elements and situations it has not seen before. This could require the entire technique to be re-engineered multiple times. Treating scene perception as a general problem for which the solution does not technically depend on the exact modules used is the key to rethinking what it means for a robot to perceive entities and changes in its environment in real-time. The focus is on determining the position of objects and a human target relative to the robot's central field of view. However, the proposed method can theoretically be extended to include depth information for path planning. The segmentation framework used could be replaced with a more efficient one. The default set of values for pixel re-assignment could be extended to include various groups of objects based on their importance or danger levels. The default list of kernels proposed could be expanded to include larger kernels for higher-resolution images. The coefficient from the dynamic kernel selection algorithm could be tweaked based on the level of pixel information required to retain. The possibilities to amend, improve and scale the proposed technique are endless. The essential takeaway from this paper is that it demonstrates a novel yet simple approach for performing scene understanding using pixel information without extensive training time. It does so primarily by using image segmentation and applying fundamental matrix mathematics to aid a robot in perceiving the angular position of entities in its surroundings. Moreover, it works irrespective of the camera's RGB resolution (Table 1) and input image size and is invariant to poor lighting conditions – indoors or outdoors. It is also functional as a standalone system that can be used as part of a vision-based surveillance system without restrictions on what can be tracked. It may be used directly on a webcam like in Fig. 3, which means it can be ported onto different platforms to be integrated with other software and algorithms.

Future work for this research includes implementing this logic into a ROS package and performing extensive evaluations for real-time robot-human following. It also

involves assessing the technique's accuracy regarding navigation in free space and the presence of multiple obstacles in the real world. In the next phase of this research, these enhancements shall be carried out and reported in detail, with supporting evidence.

References

1. Alatise, M.B., Hancke, G.P.: A review on challenges of autonomous mobile robot and sensor fusion methods. IEEE Access **8**, 39830–39846 (2020). https://doi.org/10.1109/ACCESS.2020.2975643
2. Lau, B.T.: An Improved indoor robot human-following navigation model using depth camera, active IR marker and proximity sensors fusion. Robotics **7**(1), 4 (2018). https://doi.org/10.3390/robotics7010004
3. Barber, R., Crespo, J., Gómez, C., Hernámdez, A.C., Galli, M.: Mobile robot navigation in indoor environments: geometric, topological, and semantic navigation. Appl. Mob. Robot. (2018). https://doi.org/10.5772/INTECHOPEN.79842
4. Wang, M., Su, D., Shi, L., Liu, Y., Miro, J.V.: Real-time 3D human tracking for mobile robots with multisensors. In: 2017 IEEE International Conference on Robotics and Automation, pp. 5081–5087 (2017). https://doi.org/10.1109/ICRA.2017.7989593
5. Chen, B.X., Sahdev, R., Tsotsos, J.K.: Integrating stereo vision with a CNN tracker for a person-following robot. In: Liu, M., Chen, H., Vincze, M. (eds.) ICVS 2017. LNCS, vol. 10528, pp. 300–313. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68345-4_27
6. Liu, H., Luo, J., Wu, P., Xie, S., Li, H.: People detection and tracking using RGB-D cameras for mobile robots. Int. J. Adv. Robot. Syst. **13**(5), 172988141665774 (2016). https://doi.org/10.1177/1729881416657746
7. Condés, I., Cañas, J.M.: Person following robot behavior using deep learning. In: Fuentetaja Pizán, R., García Olaya, Á., Sesmero Lorente, M.P., Iglesias Martínez, J.A., Ledezma Espino, A. (eds.) WAF 2018. AISC, vol. 855, pp. 147–161. Springer, Cham (2019). https://doi.org/10.1007/978-3-319-99885-5_11
8. Chen, E.: lFOLOr: a vision-based human-following robot, pp. 224–232 (2018). https://doi.org/10.2991/amcce-18.2018.40
9. Condés, I., Cañas, J.-M., Perdices, E.: Embedded deep learning solution for person identification and following with a robot. In: Bergasa, L.M., Ocaña, M., Barea, R., López-Guillén, E., Revenga, P. (eds.) WAF 2020. AISC, vol. 1285, pp. 291–304. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-62579-5_20
10. Nguyen, A., Tran, Q.: Autonomous navigation with mobile robots using deep learning and the robot operating system (2020). http://arxiv.org/abs/2012.02417. Accessed 18 June 2021
11. Chan, W.P., Radmard, S., Hew, Z.Q., Morris, J., Croft, E., Van der Loos, H.F.M.: Autonomous person-specific following robot (2020). http://arxiv.org/abs/2010.08017. Accessed 09 July 2021
12. Tensorflow: models/research/object_detection at master · tensorflow/models. GitHub (2021). https://github.com/tensorflow/models/tree/master/research/object_detection. Accessed 26 July 2021
13. Duan, K., Bai, S., Xie, L., Qi, H., Huang, Q., Tian, Q.: CenterNet: keypoint triplets for object detection. In: Proceedings of IEEE International Conference on Computer Vision, vol. 2019-Octob, pp. 6568–6577 (2019). https://doi.org/10.48550/arxiv.1904.08189
14. Tan, M., Pang, R., Le, Q.V.: EfficientDet: scalable and efficient object detection. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 10778–10787 (2019). https://doi.org/10.48550/arxiv.1911.09070

15. Liu, W., et al.: SSD: single shot multibox detector. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9905, pp. 21–37. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_2
16. Lu, X., Kang, X., Nishide, S., Ren, F.: Object detection based on SSD-ResNet. In: Proceedings of 2019 6th IEEE International Conference on Cloud Computing and Intelligence Systems, CCIS 2019, pp. 89–92 (2019). https://doi.org/10.1109/CCIS48116.2019.9073753
17. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans. Pattern Anal. Mach. Intell. **39**(6), 1137–1149 (2017). https://doi.org/10.1109/TPAMI.2016.2577031
18. Padilla, R., Passos, W.L., Dias, T.L.B., Netto, S.L., Da Silva, E.A.B.: A comparative analysis of object detection metrics with a companion open-source toolkit. Electron **10**(3), 279 (2021). https://doi.org/10.3390/ELECTRONICS10030279
19. He, K., Gkioxari, G., Dollar, P., Girshick, R.: Mask R-CNN, pp. 2961–2969 (2017)
20. Luiten, J., Torr, P., Leibe, B.: Video instance segmentation 2019: a winning approach for combined detection, segmentation, classification and tracking. In: 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), pp. 709–712 (2019). https://doi.org/10.1109/ICCVW.2019.00088
21. Luiten, J., Voigtlaender, P., Leibe, B.: PReMVOS: proposal-generation, refinement and merging for video object segmentation. In: Jawahar, C., Li, H., Mori, G., Schindler, K. (eds.) ACCV 2018. LNCS (LNAI and LNB), vol. 11364, pp. 565–580. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-20870-7_35
22. Liu, S., Qi, L., Qin, H., Shi, J., Jia, J.: Path aggregation network for instance segmentation. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern recognition, pp. 8759–8768 (2018). https://doi.org/10.1109/CVPR.2018.00913
23. 3D Camera Survey. ROS-Industrial (2022). https://rosindustrial.org/3d-camera-survey. Accessed 12 July 2022
24. Tee Kit Tsun, M., Bee Theng, L., Siswoyo Jo, H., Lun Lau, S.: Proposing a sensor fusion technique utilizing depth and ranging sensors for combined human following and indoor robot navigation (2016). http://dx.doi.org/10.1145/3033288.3033345. Accessed 17 May 2021
25. Quaternion Message. ROS Documentation (2022). http://docs.ros.org/en/noetic/api/geometry_msgs/html/msg/Quaternion.html. Accessed 16 May 2022
26. Gohlke, C.: Homogeneous transformation matrices and quaternions. Laboratory for Fluorescence Dynamics. University of California, Irvine, California (2021). https://github.com/cgohlke/transformations/. Accessed 16 May 2022
27. Bharati, P., Pramanik, A.: deep learning techniques—R-CNN to mask R-CNN: a survey. In: Das, A., Nayak, J., Naik, B., Pati, S., Pelusi, D. (eds.) Computational Intelligence in Pattern Recognition. Advances in Intelligent Systems and Computing, vol. 999, pp.657–668. Springer, Singapore (2020). https://doi.org/10.1007/978-981-13-9042-5_56
28. Release Mask R-CNN 2.1 · matterport/Mask_RCNN. https://github.com/matterport/Mask_RCNN/releases/tag/v2.1. Accessed 15 May 2022
29. Abdulla, W.: Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow. GitHub (2017). https://github.com/matterport/Mask_RCNN. Accessed 15 May 2022
30. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2016-Decem, pp. 770–778 (2015). https://arxiv.org/abs/1512.03385v1. Accessed 22 July 2021
31. Chauhan, R., Ghanshala, K.K., Joshi, R.: Convolutional neural network (CNN) for image detection and recognition. In: 2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC), pp. 278–282 (2018). https://doi.org/10.1109/ICSCCC.2018.8703316

32. Ran, T., Yuan, L., Zhang, J.B.: Scene perception based visual navigation of mobile robot in indoor environment. ISA Trans. **109**, 389 (2021). https://doi.org/10.1016/J.ISATRA.2020.10.023
33. A. Uçar, Y. Demir, and C. Güzeliş: Object recognition and detection with deep learning for autonomous driving applications, vol. 93, no. 9, pp. 759–769 (2017). https://doi.org/10.1177/0037549717709932
34. Gupta, A., Anpalagan, A., Guan, L., Khwaja, A.S.: Deep learning for object detection and scene perception in self-driving cars: survey, challenges, and open issues. Array **10**, 100057 (2021). https://doi.org/10.1016/J.ARRAY.2021.100057
35. Du, Q., et al.: Deep learning-based object detection and scene perception under bad weather conditions. Electron. **11**(4), 563 (2022). https://doi.org/10.3390/ELECTRONICS11040563
36. Wang, L., et al.: Multi-view fusion-based 3D object detection for robot indoor scene perception. Sensors **19**(19), 4092 (2019). https://doi.org/10.3390/S19194092
37. Chen, L., Yang, Z., Ma, J., Luo, Z.: Driving scene perception network: real-time joint detection, depth estimation and semantic segmentation. In: 2018 IEEE Winter Conference on Applications of Computer Vision, vol. 2018-Janua, pp. 1283–1291 (2018). https://doi.org/10.1109/WACV.2018.00145
38. Kaur, B., Bhattacharya, J.: Scene perception system for visually impaired based on object detection and classification using multimodal deep convolutional neural network. J. Electron. Imaging **28**(01), 1 (2019). https://doi.org/10.1117/1.JEI.28.1.013031
39. Brunetti, A., Buongiorno, D., Trotta, G.F., Bevilacqua, V.: Computer vision and deep learning techniques for pedestrian detection and tracking: a survey. Neurocomputing **300**, 17–33 (2018). https://doi.org/10.1016/j.neucom.2018.01.092