A Data-Driven Algorithm Based on YOLOv8 for Object Detection, Tracking, Distance Estimation, and Size Measurement in Stereo Vision Systems

Amirhossein Dadashzadeh Taromi[1] Sajad Haghzad Klidbary[2*]

[1] Department of Engineering, University of Zanjan, Zanjan, Iran. shervin.dadashzade7988@gmail.com

[2] Department of Engineering, Faculty of Computer Engineering, University of Zanjan, Zanjan, Iran. s.haghzad@znu.ac.ir

[*] Correspondence: s.haghzad@znu.ac.ir

***Abstract*—Stereo vision systems, inspired by human visual perception, can infer depth from images using two cameras. This study emphasizes explicitly the measurement of distance and the size of objects of interest, both critical components for many autonomous and robotic systems. As DL and ML continue to advance within computer vision applications, adapting traditional algorithms into modern machine learning-based solutions promises heightened accuracy. Our investigation uses the YOLOv8 object detection model to identify objects of interest. Our algorithm involves a data-driven approach toward distance and size measurement by training a neural network to predict the distance and size of objects within a stereo vision system. The key innovation of this method lies in the neural network's ability to learn and model the relationship between disparity and distance, as well as width and height, through real-world data collection. The proposed algorithm eliminates the heavy reliance on mathematical formulas typically tied to calibration processes. In our experiments, conducted at distances ranging from 50 to 200 centimeters, our proposed algorithm showcased accurate performance. It achieved distance measurements with an accuracy of up to 99.99% in select cases and maintained the mean accuracy of 98.15% for distance, 92.87% for width, and 93.92% for height measurements.**

keywords: Distance Measurement, Size Measurement, Machine Learning, Deep Learning, Neural Networks, YOLOv8, Stereo Vision

I. Introduction

Object distance estimation and size measurement are essential for most autonomous vehicles and robot systems. Surrounding objects' type and distance are critical to performing localization, navigation, and obstacle detection of autonomous systems [1], and additional information, such as the size of the objects, can improve their accuracy and performance. Therefore, detecting surrounding objects and accurately measuring their distance and size in real-time is a critical and challenging task in many autonomous systems [2]. Size and distance information, in addition to their vital role in autonomous systems, hold significant importance in various fields of study. One such example is the domain of aquaculture engineering, where the measurement of fish length serves as an essential monitoring parameter [3]. Due to the importance of this topic, researchers have conducted many studies to calculate the distance and size of objects. These studies are in two primary categories: active methods and passive methods.

Active methods mainly focus on calculating the distance by emitting electromagnetic waves or using laser beams. RADAR(Radio Detection and Ranging) sensors, for example, emit a radio-frequency wave, and by considering the time it took for the wave to hit and get reflected by the target object, they can calculate the distance based on the wave's speed of the propagation in the air [4]. Ultrasonic sensors use the same methodology as the RADAR sensors for distance measurement, but they generate ultrasonic wave bursts. On the other hand, LIDAR (Light Detection and Ranging) sensors utilize infrared laser beams. These sensors analyze the reflected beams from a target object to measure its distance [5]. LIDAR sensors are more accurate than RADAR sensors based on the study of [6]. They used two different SLAM(Simultaneous Localization and Mapping) algorithms to compare the accuracy of LIDAR and RADAR sensors. They found that the mean displacement in position when using the RADAR sensor was less than 0.037 meters, compared to 0.011 meters for the LIDAR. RADAR and ultrasonic sensors have some notable drawbacks as well. One of the primary concerns is the potential confusion for echoes from previous or subsequent pulses and interference from other sensors that can lead to noise. Furthermore, the precision of distance measurement achieved by these sensors is typically constrained within the range of 1 to 4 meters [2]. Meanwhile, the higher cost associated with LIDAR sensors, combined with the inherent potential for vision-based measurement systems to be automated for accurate measurements and proficient management of systematic errors, establishes the utilization of vision systems as a more desirable choice [7].

Conversely, passive methods mainly focus on using vision systems to estimate distance and measure the size of objects. The camera manufacturing industry has made significant advancements lately, drastically reducing the price of these devices. Nowadays, most robotic and autonomous systems come equipped with at least one camera, making using vision sensors as a measurement instrument a preferred option. Monocular vision systems consist of one camera, and many studies have focused on estimating the distance of particular objects using monocular vision systems. For instance, [8] introduced an algorithm focused on detecting, avoiding, and measuring distances using a monovision system. Their approach involved employing calibration techniques alongside extracting and matching key points from two successive frames by utilizing SIFT and SURF algorithms. Then, they assess the vehicle's distance from a frontal obstacle through variations in those matched points. Moreover, [9] proposed a monovision system that combines instance segmentation and camera focal length to measure the distance between a car and the vehicles ahead. Their system consists of three main steps. Initially, it recognizes the location of the cars within the image. Then, it classifies the car based on its type using a classification model trained on the CompCars dataset [10]. Then, they use an instance segmentation model trained on the Cityscapes dataset [11] to segment the car. Finally, the third stage focuses on distance calculation for the identified cars. This computation establishes the correlation between size-related information pertinent to different car types and their corresponding mask values. Also, [12] proposed an algorithm utilizing Hugh transforms for image processing alongside the relative

size of a target object to measure the distance in a monocular vision system. As another example, [13] introduced a system for measuring the distance between a person and a camera, employing a single camera and focusing on eye distance as a metric. This system determines the distance by analyzing the fluctuations in eye distance presented in pixels, corresponding to alterations in the camera-to-person distance. The main challenge for distance measurement in monocular vision systems lies in the intricate nature of the employed algorithms, which encompasses classifying object types and aligning the actual dimensions of objects with their dimensions in images across different frames. This complexity is particularly evident in scenarios involving overlapping objects.

While many research studies have focused on distance measurement through monocular vision systems, many works advocate adopting stereo-vision systems as a more accurate method. Stereo vision systems endeavor to replicate human visual perception by utilizing a pair of cameras to infer depth information from images. This method replicates the human vision system for a more comprehensive depth understanding. In the work of [14], they presented an algorithm implemented in C++ for object distance estimation and size measurement for a stereo vision system. For object detection, they utilized background subtraction to localize the object, besides a database of different classes of objects with their corresponding width and height, to classify the type of object based on the measured size. They used a mathematical equation based on the disparity of objects in left and right frames for distance estimation and a linear model for size measurement. Their proposed algorithm successfully estimated objects' distance with a precision of $\pm$ 25cm, and their reported object size measurement error is not more than $\pm$ 3cm, which is reasonably accurate in the 3 to 10-meter range. Researchers reported the algorithm's mean processing rate as 15 frames per second on a single-core Pentium 4 3.0GHz processor. Moreover, the [2] study has focused on distance measurement of frontal cars using stereo vision on autonomous vehicles. Their approach involves several key steps. In the object detection phase, edge detection and cross-correlation [15] techniques identify potential vehicle positions in images. Extracting features from these hypothesis boxes involves using the two-dimensional(2D) discrete wavelet transform [16] method. These extracted features are then input into an AdaBoost classifier, distinguishing between vehicles and non-vehicles. Interestingly, the researchers apply their object detection algorithm to images from one camera, and through cross-correlation, they pinpoint and match corresponding vehicles in the images from the other camera, effectively reducing computational time. In the next step, they calculate the distance using a mathematical equation based on the position of vehicles in the two images, view angles, and the distance between two cameras. Furthermore, in the study of [17], researchers introduced a cost-effective computational algorithm inspired by bacterial behavior for stereo vision systems. This algorithm was designed for real-time obstacle detection and distance measurement within the range of 0.3 to 2 meters. The algorithm demonstrated competence in obstacle coverage and distance measurement, aligning with the achievements observed in related studies within this field. A different study [18] investigated distance measurement in stereo vision systems. Researchers implemented their suggested algorithm in Python. They also utilized background subtraction to localize objects of interest and used the disparity values in the depth map instead of calculating the disparity only for the particular target object. In the distance estimation phase, researchers used the same formula presented in the study of [14] for distance measurement. They achieved an average accuracy of 98.19% in the 60-200cm range and a processing time of 0.355 s (roughly three frames per second) on an Intel Core i7 CPU running at frequencies of 2.70 and 2.90GHz. In a recent study of [19], the application of stereo vision systems in real-time fire analysis was explored. Their algorithm, specifically designed for fire assessment, incorporated several crucial components. Firstly, they employed YOLOv5 for fire detection, allowing the system to identify and locate fires efficiently. To measure distances and heights accurately within the dynamic range of 50 to 80 centimeters, the algorithm utilized bounding boxes alongside the Semi-Global Block Matching (SGBM) technique, which generated depth maps. Furthermore, the study addressed the critical parameter of fire power measurement. To accomplish this, a VGG16 model was trained to estimate the Heat Release Rate (HRR) using data from the NIST dataset. One of the notable challenges in fire-related measurements is the ever-changing nature of fire flames. Despite this complexity, the reported distance measurement accuracy of approximately 5% was deemed adequate for the specific application, highlighting the algorithm's practical utility in real-time fire assessment scenarios.

In the existing research landscape, there are gaps that our work seeks to address. Firstly, background subtraction and traditional object detectors were the main algorithms for the object detection phase in reviewed studies. The main inconvenience with the background subtraction is its poor performance in moving camera scenarios. Apart from this, traditional object detection involving manual feature extraction is outperformed by modern object detectors based on DL, whether in accuracy or the number of classes they can detect. Secondly, all the reviewed studies try to calculate the distance or size of objects based on disparity using a mathematical equation that relies on the camera calibration process, which means any error in this phase would be propagated and directly impact the algorithm's accuracy. Considering the recent advancements in ML and DL, especially in computer vision, object detection models learn to detect and distinguish vast numbers of objects even with similar features. In addition, recent ML algorithms can learn the complex relationship between numbers, making them suitable to provide a realistic model from collected data. This paper proposes an algorithm adopted to recent ML and DL advances for distance and size measurement in stereo vision systems. In general, we can underline our contribution to the field as follows:

1) We propose a general and customizable algorithm for object detection, tracking, distance estimation, and size measurement for objects of interest in stereo vision systems.
2) We present a novel approach toward distance and size measurement by training a neural network(NN) to model the relationship between distance, size of objects, and the disparity value based on real-world collected data for a reference object.
3) We investigate the generalization of the neural network and its performance and accuracy on the obstacles it has not seen before in the 50-200 centimeters range.
4) We integrate YOLOv8, a uniform end-to-end object detection and tracking model, into our algorithm.
5) We also prove that our algorithm can perform and be used in scenarios involving uncalibrated cameras, so we consider camera calibration and stereo rectification an optional step in our proposed algorithm.

The rest of paper is constructed as follow, section 2 briefly reviews foundational concepts in stereo vision, ML, and object detection required to understand the algorithm. Section 3 describes the algorithm's steps, including stereo calibration and rectification, stereo matching, and distance and size measurement. In section 4, we assess the algorithm's accuracy by experimenting with real-world scenarios for four obstacles and comparing our algorithm performance with other studies in the literature. Finally, in the last section, we conclude and consider future improvements and studies around this work.

II. Foundational Concepts

A. Camera Model

A camera model is a mathematical representation of the projection process for a 3D point in real-world coordinates onto a 2D image plane formed inside a camera. In this study, we use a Pinhole camera model to represent the image formation of the camera. Based on the pinhole model, the projection of a 3D point P is the 2D point p resulting from the intersection of the CP line with the image plane, where C is the camera's optical center. Finding the projection of point P is called the forward projection, visible in Fig. 1.

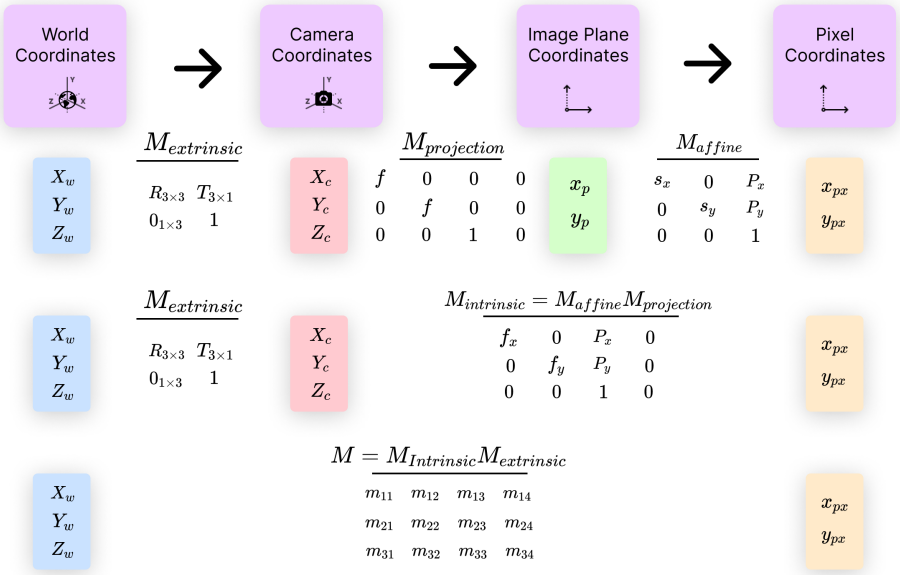

Fig. 1: Forward projection process, describing the formation and role of each matrix.

A 3D point P is defined by its coordinates (X_w, Y_w, Z_w) to the world coordinate system. A series of transformations are applied to project 3D point P onto a 2D point p. Initially, the extrinsic parameters are responsible for mapping the world coordinates of point P to the camera coordinate system, resulting in coordinates (X_c, Y_c, Z_c). The extrinsic parameters include a $R_{3\times3}$ rotation matrix, which represents the rotation of the camera, and a $T_{3\times1}$ translation vector, which indicates the transition from the world coordinate system to the camera coordinate system with the camera's center(C) as the origin.

Subsequently, point P is projected onto the image plane using the projection matrix $M_{projection}$, which incorporates the camera's focal length. At this stage, the projected 3D point P takes the form of (x_p, y_p) on the image plane. The affine transformation matrix M_{affine} is employed to convert these coordinates from a measurement metric to pixel units. Also, we consider the $M_{intrinsic} = M_{projection}M_{affine}$ as the intrinsic parameters of the camera that are unique and constant for each camera, where $M_{extrinsic}$ or external parameters of camera can change based on world coordinate system.

B. Lens distortion

Lens distortion occurs when a camera lens fails to represent objects' shape and size in an image accurately. In the studies of [20] and [21], researchers have investigated and modeled the two main types of lens distortion: radial and tangential. Radial distortion happens when straight lines in a photo appear to curve, usually towards the edges of the image, and the lens form causes this type of distortion. Whereas tangential distortion occurs when the camera's lens and image sensor are not perfectly aligned, causing objects in the image to look stretched or compressed in certain directions. Considering the (x, y) as deformed point, the corrected point (x_{cor}, y_{cor}) for radial distortion can be calculated using the equation 1:

$$\begin{pmatrix} x_{cor} \\ y_{cor} \end{pmatrix} = \begin{pmatrix} x(1 + k_1r^2 + k_2r^4 + k_3r^6) \\ y(1 + k_1r^2 + k_2r^4 + k_3r^6) \end{pmatrix} \quad (1)$$

Furthermore, for the correction of tangential distortion, the formula presented in equation 2 is used:

$$\begin{pmatrix} x_{cor} \\ y_{cor} \end{pmatrix} = \begin{pmatrix} x + (2p_1xy + p_2(r^2 + 2x^2)) \\ y + (p_1(r^2 + 2y^2) + 2p_2xy) \end{pmatrix} \quad (2)$$

Where in both equations of 1 and 2 the k_1, k_2, k_3 and p_1, p_2 are the distortion coefficients, and camera calibration process determines their values.

C. Stereo rectification

Calibrating the stereo setup is a fundamental step in achieving stereo rectification, which involves the transformation of the left and right camera frames to make them parallel and align all epipolar lines along the horizontal axis of their respective images. Stereo rectification simplifies the stereo-matching process significantly. Instead of searching for corresponding points across the entire right image frame, we can now confine our search to a single horizontal scanline or even a patch of pixels in terms of our proposed algorithm. This process reduces computational complexity and enhances the accuracy of disparity estimation, as it enforces the epipolar constraint, ensuring that corresponding points in the rectified images lie on the same horizontal line.

D. Artificial Neural Networks

Artificial Neural Networks (ANNs), inspired by the mechanism of the human brain, aim to learn from data and model a function to provide the target data. These networks consist of neurons, each with weights and a bias, that process input signals to produce the neuron's output. Layers are building blocks of NNs and are the groups of neurons. NNs typically comprise three main layers: the input, output, and hidden layers. Hidden layers are the layers between the input and output layers, and there can be zero or more hidden layers in an NN. Recent advancements in ML and DL have led to various types and architectures of NNs, including Feedforward Neural Networks(FNNs), Recurrent Neural Networks(RNNs), Convolutional Neural Networks(CNNs), Transformers, and more, making them suitable for diverse use cases and data types.

Fully Connected NNs(FCNs), known as feedforward NNs, are fundamental forms of ANNs. In such networks, there are connections between neurons in one layer and all neurons in the neighboring layers. Neurons in NNs commonly utilize activation functions on their output signals, allowing networks to learn non-linear functions.

The learning process in involves backpropagation and updating weights using the Gradient Descent(GD) algorithm. Typically, an NN starts with randomly initialized weights. Using a loss function to

quantify the error in the network's output, backpropagation and the GD algorithm determine the adjustments required for the weights and biases in various layers. This process continues until the NN effectively represents the train data.

E. Object Detection

Object detection involves the localization and classification of objects within images. Object detection algorithms allow us to identify and outline objects using bounding boxes while specifying their respective classes. Since the 1960s, companies have applied this technology to various tasks, such as character pattern recognition in office automation and the assembly and verification of different components across industries. Traditional object detection methods heavily depend on manual feature extraction, template matching, and sliding window techniques. However, these approaches come with significant limitations. In the case of template matching and sliding windows, errors in detection and reduced accuracy can occur when objects vary in pose, size, and scale or when occlusions are present. While manual feature extraction can yield accurate results for specific object categories, it is challenging to generalize this approach to detect diverse object classes due to the difficulty of manually identifying standard features that distinguish different types of objects. Hence, a key challenge faced by algorithms like the Viola-Jones detector [22] and object detection methods utilizing features extracted from histograms of oriented gradients (HOG) [23] is achieving generalization across a diverse range of object classes.

Technological and computational power advances led to the emergence of large datasets that enabled NNs to find their path in image classification and object detection. ImageNet [24] is one of these datasets that started an annual software contest, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). The AlexNet [25], a CNN image classification network inspired by LeNet [26], in 2012 marked the starting of the modern era of object detection using NNs and DL.

The next turning point was the Region-Based CNN (R-CNN) [27] was introduced in 2013. This model uses a selective search algorithm to extract 2,000 region proposals. Then, a CNN classifies those regions, and a regression algorithm refines the bounding boxes. The proposed model was accurate but slow in action and impractical for real-time applications, so the different models based on R-CNN were proposed and tried to fix this problem, such as Fast R-CNN [28] and Faster R-CNN [29].

The You Only Look Once (YOLO) model, as implied by its name, is a single CNN designed to predict classes and bounding boxes in a single pass simultaneously. The study of [30], for the first time, introduced the YOLO model in the object detection field in 2015. This model operates by dividing the input image into evenly-sized cells, each responsible for predicting a certain number of bounding boxes. Boxes with a low probability of containing an object are discarded, while those with higher confidence are refined to more accurately enclose the objects of interest. YOLO's approach of processing the input image only once through the network results in reduced computational demands, making it suitable for real-time applications.

The initial version of YOLO had certain limitations. It imposed strict spatial constraints on bounding box predictions, limiting its ability to detect nearby objects effectively. Additionally, when small objects appeared in groups, prediction accuracy suffered. Several subsequent versions of YOLO have been developed to address these challenges and enhance model accuracy and performance, including YOLOv2, YOLOv3, YOLOv4, YOLOv5, YOLOv6, YOLOv7, and YOLOv8.

YOLOv8 is the latest iteration of the YOLO series and represents the state-of-the-art model developed by Ultralytics. Ultralytics has created a comprehensive Python library for this model, supporting a wide range of tasks, including image classification, object detection, object tracking, instance segmentation, and pose detection. As illustrated in Fig. 2, YOLOv8 demonstrates superior performance in terms of accuracy compared to its predecessors while maintaining a smaller number of parameters and faster inference times. In object tracking, Ultralytics has also integrated the BoT-SORT [31] and ByteTrack [32] algorithms with the YOLOv8 model, enabling efficient and effective object-tracking capabilities.

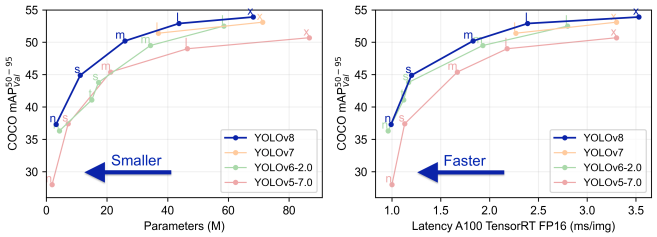

Fig. 2: Comparison of YOLO different versions developed by Ultralytics in terms of mAP, parameters, and latency on the COCO dataset. Retrieved from https://github.com/ultralytics/ultralytics

YOLOv8 stands out as an anchor-free model, meaning it directly predicts the center of an object without relying on offsets from predefined anchor boxes. This innovative approach reduces the number of box predictions, significantly accelerating the Non-Maximum Suppression (NMS) process. This intricate post-processing step refines proposed bounding boxes after inference. YOLOv8 also incorporates Mosaic augmentation during its training routine, a technique that involves combining four images to diversify the model's learning experience. This augmentation strategy enhances the model's robustness by presenting objects in new locations, scenarios involving partial occlusion, and against varying surrounding pixels.

III. Proposed Algorithm

A. Stereo setup, hardware and software specifications

In the context of our study, the stereo configuration encompasses the utilization of two Logitech C270 webcams positioned with a baseline displacement of 7.5 centimeters. We securely affix these webcams to a supportive staff structure. Operating at a resolution of 720p, they capture video at 30 frames per second(FPS). Our proposed algorithm is implemented on a laptop system, using an Intel Core i5 7th generation CPU with 12GB of RAM, clocking at 3.0GHz. This system operates on the Ubuntu 20.04 operating system, providing a robust foundation for our analyses. Moreover, our proposed algorithm used Python 3.8.10, OpenCV 4.7, and PyTorch 1.10 as the primary programming language and libraries.

The proposed algorithm presented in this study comprises a systematic process depicted in Fig. 3, which can be summarized into two optional and three mandatory steps:

1) **Camera Calibration (Optional)**: This initial step involves the calibration of each camera within the system and the stereo calibration of the overall stereo setup.
2) **Stereo Rectification (Optional)**: In this phase, the algorithm rectifies and eliminates distortions in the stereo vision system's left and right frames.
3) **Object Detection and Tracking**: The left frame acquired from the camera undergoes processing utilizing the YOLOv8 model, resulting in the extraction of bounding boxes and unique identifiers for tracking objects.
4) **Stereo Matching**: In this step, the algorithm employs template matching techniques to identify corresponding objects in the right frame. This process facilitates the calculation of disparity values for each matched object pair.
5) **Distance and Size Measurement**: Leveraging the computed disparity values, an NN is employed to predict the distances and sizes of the detected objects.

In the ensuing sections, we shall provide an exhaustive review and elaborate on these algorithmic steps.

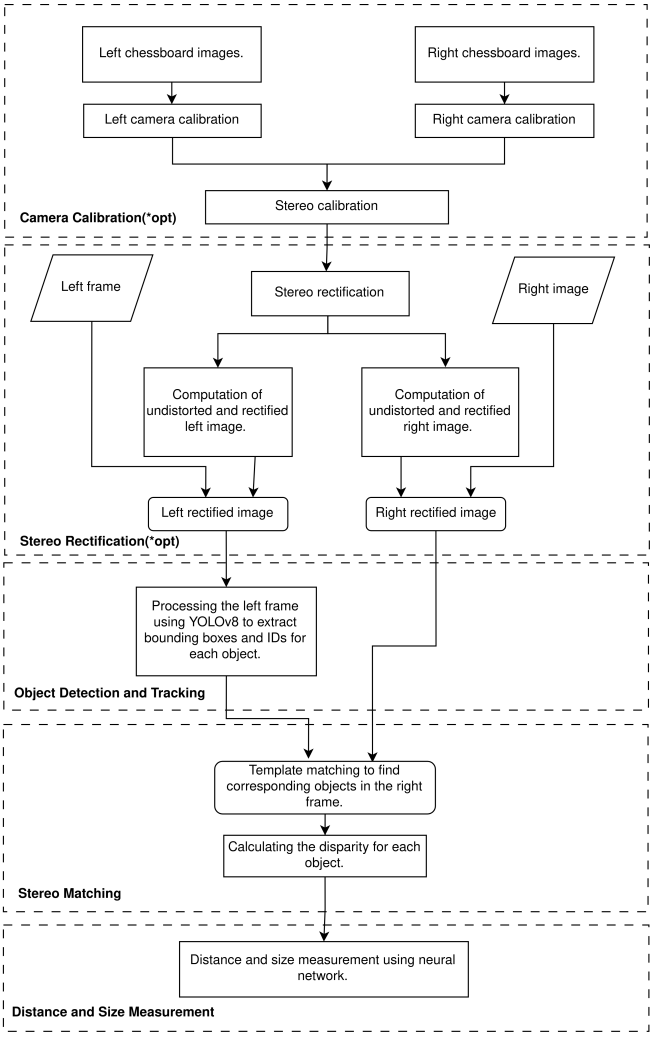

Fig. 3: The proposed algorithm flowchart shows the 5 steps in our algorithm. The *opt symbol indicates that the camera calibration and stereo rectification steps are optional.

B. Camera Calibration and Stereo Rectification

We aimed to evaluate our proposed algorithm's accuracy and performance under two distinct scenarios:

1) Involving calibrated cameras with stereo-rectified and distortion-corrected images.
2) Employing uncalibrated cameras without any correction or rectification.

That is why we considered the stereo calibration and rectification process in our proposed algorithm flowchart in Fig. 3 optional. For the calibration process, we utilized a chessboard pattern featuring a grid of 9 by 7 corners, each spaced 2.5 centimeters apart. The camera calibration process in this study comprises two primary steps. Initially, we calibrate each camera, capturing images from various viewpoints and orientations of a chessboard. This comprehensive coverage facilitates the determination of intrinsic parameters and distortion coefficients unique to each camera. Subsequently, we execute stereo calibration by simultaneously capturing chessboard images with both cameras. Leveraging the intrinsic parameters established in the prior step, we calibrate our stereo system and ascertain the extrinsic parameters that define the relative positioning of the two cameras. We collected 20 images for each camera in the single-view camera calibration step. These images yielded calibration Root Mean Square Error(RMSE) values of 0.173 and 0.149 for the left and right cameras, respectively. Additionally, we acquired 20 pairs of images for the stereo calibration phase, resulting in a calibration RMSE of 0.42. Fig. 4 showcases sample images and the impact of the distortion correction and stereo rectification processes on the images.

C. Object Detection and Tracking

In this section, we initially employed the YOLOv8 model and its implemented BoT-SORT tracking algorithm to detect and track objects of interest and extract bounding boxes within the left frame. To facilitate our goal of proposing a real-time distance and size measurement algorithm, we opted for YOLOv8n, the most lightweight variant of YOLOv8 with 3.2 million parameters. This choice ensures high-performance and real-time object detection even on CPU devices. Notably, for the scope of this study, we refrained from training the YOLO model on a custom dataset and instead utilized the default pretrained model based on the COCO [33] dataset and can detect 80 various classes. Additionally, to expedite processing times, we captured frames from the cameras at a size of 640x640 pixels but fed 480x480 pixel images into our YOLO model. While our decision to feed 480x480 images into our YOLO model significantly enhances processing speed, it is essential to acknowledge that this reduction in image resolution may have a marginal effect on the precision of object detection.

D. Stereo Matching

In the next step, we aimed to locate the objects initially detected in the left frame within the right frame. We utilized a template matching technique to achieve this, employing the Normalized Cross-Correlation Coefficient as the similarity metric. The similarity metric equation is presented in following. We set a minimum confidence threshold of $>=50\%$ to ensure that the algorithm would pick the best-fit matches. Our experiments demonstrated that this confidence level allows the system to detect the desired matches effectively while preserving all objects with minimum losses on detected objects.

$$R(x,y) = \frac{\sum_{x',y'}(T'(x',y') \cdot I'(x+x',y+y'))}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}} \tag{3}$$

We achieved significant time savings by opting for template matching instead of running the YOLOv8 model on the right frame. We achieved significant time savings by opting for template matching instead of running the YOLOv8 model on the right frame. This improvement increased FPS from 5 to 9. Fig. 5 visually represents the performance gains by comparing the FPS in two methods in the chart.

After identifying bounding boxes in both the left and right frames, we calculate the disparity for each object. Each bounding box is represented by two points, p_1 pointing to the upper-left corner and p_2 to the bottom-right corner of the bounding box. For instance, the left bounding box of object i is represented as $BBox_i^{left} = \left\langle (x_{min}^{left}, y_{min}^{left}), (x_{max}^{left}, y_{max}^{left}) \right\rangle$. We employ equation 4 to calculate the disparity for each object.

$$d_i = |X_{min}^{left} - X_{min}^{right}| \tag{4}$$

E. Object Distance and Size measurement

The novelty of this research lies in the utilization of NNs for distance estimation and size measurement, in contrast to conventional methods that rely on mathematical formulas dependent on calibration processes. In prior approaches, any error in the calibration process would directly impact the accuracy of object distance and size measurements. However, by adopting NNs, we enable the network to autonomously learn the essential parameters for distance and size calculation based on disparity information. Previous studies [18] and

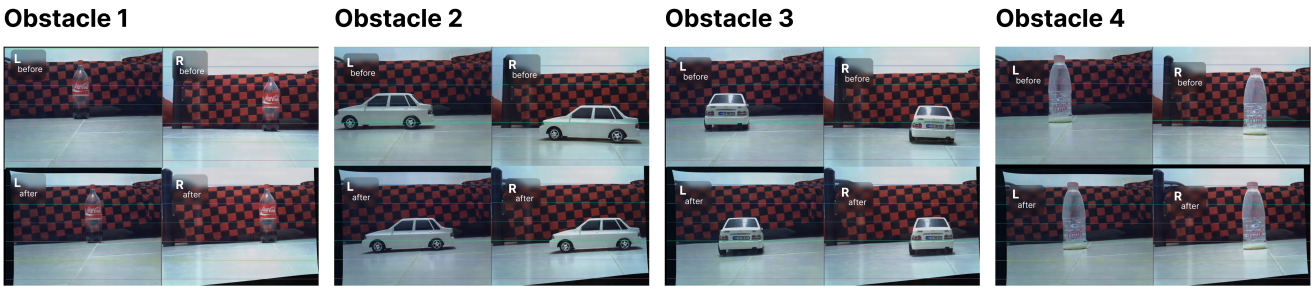

Fig. 4: Illustrating the effects of stereo rectification and distortion elimination on four obstacles in our stereo vision setup. The first row depicts the scene before rectification and distortion removal, while the second row displays the results after applying these steps.

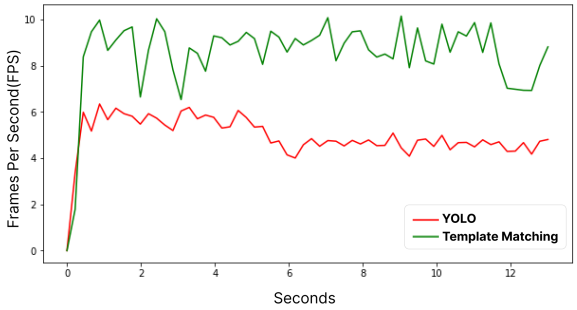

Fig. 5: Comparison of FPS difference between Template Matching method and using YOLO to process the right frame.

[14] have modeled the distance by disparity values using the equation 5. In this equation, "f" represents the focal length obtained during the camera calibration process, while "B" denotes the separation distance between the two cameras.

$$Distance = \frac{f \times B}{d} \tag{5}$$

Moreover, for size measurement, based on the study of [14], we have established an equation 6, in which w and h denote the width and height of the detected bounding boxes, respectively. Additionally, λ_w and λ_h are coefficients derived from disparity values that convert w and h from pixels to our measurement unit, which is centimeters. Their study modeled λ_w and λ_h as linear functions. Nevertheless, as we will demonstrate in the following section, based on our experiments and collected data, it becomes evident that a linear model may not precisely replicate the behavior of these functions.

$$\begin{aligned} W &= \lambda_w(d) \times w \\ H &= \lambda_h(d) \times h \end{aligned} \tag{6}$$

In contrast, this study employs a Multi-Layer Perceptron (MLP) to measure object distance and size within our stereo vision system. Instead of relying on the previously mentioned equations, we have developed an NN capable of simultaneously calculating the distance, λ_w, and λ_h based on the disparity and values.

$$Distance, \lambda_w, \lambda_h = f_{MLP}(d) \tag{7}$$

F. Training the neural network

As mentioned in the previous section, we employ an MLP to calculate distance, λ_w, and λ_h from the disparity values. The NN, illustrated in Fig. 7, consists of an input layer with one neuron, two hidden layers with 16 neurons each, utilizing a ReLU activation function [34], and an output layer with three neurons responsible to output the distance, λ_w, and λ_h respectively.

To train this NN, we undertook a data collection process. For this purpose, we selected an object as the reference with known width and height and gathered disparity, width, and height measurements of its bounding boxes in pixels at various distances. Our goal was to measure the distance and size of objects within 50 to 200 centimeters. Accordingly, we collected data for our reference object at distances of 50, 65, 80, 95, 110, 125, 155, 170, 185, and 200 centimeters. To determine λ_w and λ_h, we computed the ratios of the actual width and height of the object in centimeters to their corresponding bounding box widths and heights, respectively ($\lambda_W = \frac{W}{w}$ and $\lambda_h = \frac{H}{h}$ based on equation 6). Fig. 6 visualizes our collected data as scatter points. To train the NN using the collected data, we used Mean Squared Error(MSE) as the loss function, and for the optimizer, we used the Adam optimizer with a learning rate of 0.001. Furthermore, Fig. 8 displays the loss function values for each epoch. The NN has successfully learned to calculate the distance, λ_W, and λ_h for the reference object, as depicted in Fig. 6. In the upcoming section, we investigate the generalization of our NN to other obstacles and conduct a comparative analysis of our proposed algorithm's accuracy in distance estimation and size measurement tasks for objects of interest. This comparison extends to other studies within this field, allowing us to assess the algorithm's performance against existing research.

IV. Results and Comparison

As previously mentioned, in the initial step, we compare the impact of calibration within our proposed algorithm to demonstrate its effectiveness, even without a calibration and rectification step. We conducted 65 experiments for each scenario (with and without calibration) at distances ranging from 50 to 200 centimeters, using four obstacles to measure the accuracy of our proposed algorithm. As demonstrated in Table I, calibrating cameras and stereo rectifying the frames resulted in a mean accuracy of 95.35%. Conversely, without calibrating the cameras, we achieved a comparable accuracy of 94.98%, which closely matches the accuracy achieved with calibrated cameras. Interestingly, our proposed algorithm achieved higher accuracy in distance and width measurement tasks when cameras were uncalibrated. However, it showed slightly lower height measurement accuracy than the calibrated camera scenario. This paper extends the analysis by comparing the uncalibrated results to other studies within this field.

TABLE I: Effect of stereo rectification and camera calibration on proposed algorithm.

Calibration Status	Measurement Accuracy(%)			
	Distance	Width	Height	Mean
Without Calibration	**98.15**	**92.87**	93.92	94.98
With Calibration	97.91	92.03	**96.13**	**95.35**

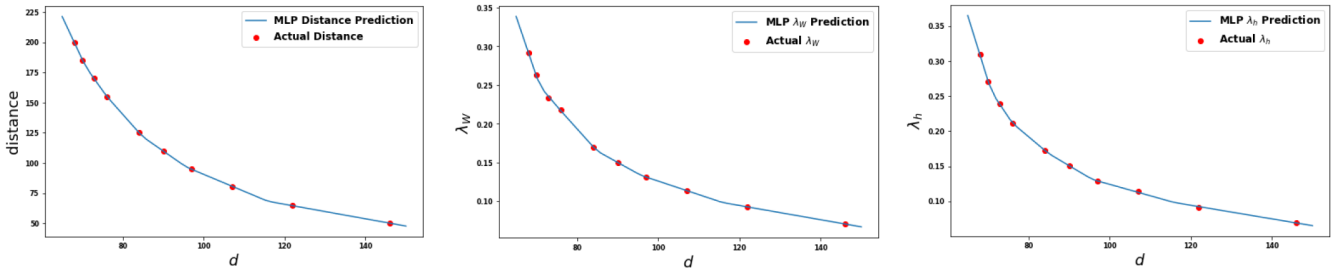

Fig. 6: Comparison of predictions from the proposed NN versus collected data in an uncalibrated scenario.

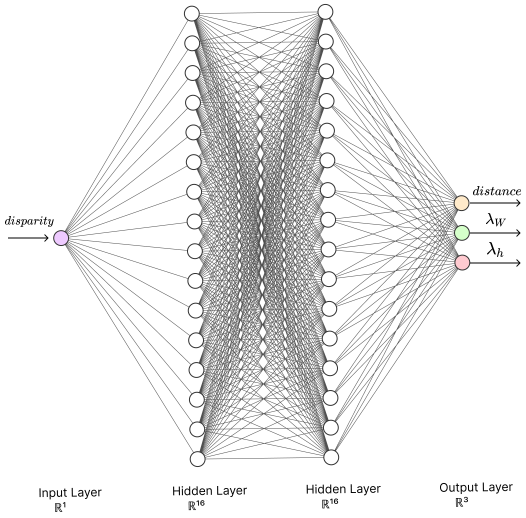

Fig. 7: The proposed NN architecture, tailored for the computation of distance, λ_w, and λ_h based on disparity value.

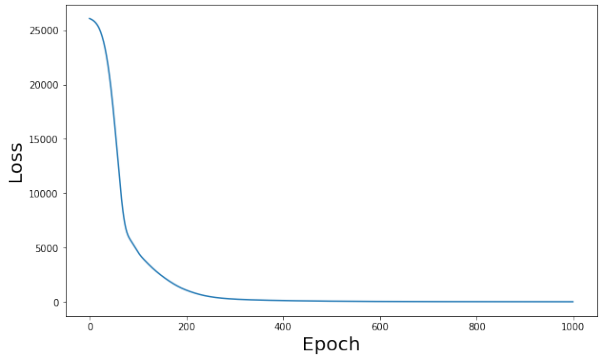

Fig. 8: Loss function plot during the training session for 1000 epochs.

A. Distance measurement accuracy

Table II presents the errors obtained for four obstacles ranging from 50 to 200 centimeters from the camera in the distance measurement task. The actual distances were measured using tape, and the error presented in the table represents the percentage difference between the predicted and actual distances. Notably, there were instances where our object detection did not successfully detect obstacles; for example, in the 180 to 200-centimeter range, YOLO was unable to detect obstacle 3. As observed in the table, the maximum error in distance measurement occurred at 50 centimeters, with a mean error of 5.076%. In comparison, the minimum mean error was recorded at 180 centimeters, with a value of 0.716%. The average mean error across our experiments is 1.842%, resulting in an accuracy of 98.15% in distance measurement, as depicted in Table I.

TABLE II: Error obtained for distance measurement without calibration.

Actual distance (cm)	Error(%)				
	Obstacle 1	Obstacle 2	Obstacle 3	Obstacle 4	Mean error(%)
50	-	0.772	7.761	6.694	5.076
60	1.312	0.423	4.021	1.355	1.778
70	1.607	0.079	6.822	1.765	2.568
80	0.630	1.583	6.746	2.320	2.820
90	2.493	0.130	2.752	0.130	1.376
100	2.788	0.327	2.133	0.327	1.394
110	2.678	0.441	4.032	0.441	1.898
120	2.586	0.536	2.540	0.536	1.550
130	2.164	0.766	3.695	2.164	2.197
140	0.991	0.991	1.729	0.991	1.176
150	1.244	0.025	3.129	0.025	1.106
160	0.001	0.001	3.314	0.001	0.829
170	0.357	0.357	3.478	0.357	1.137
180	0.797	0.676	-	0.676	0.716
190	1.831	0.961	-	1.831	1.541
200	4.087	1.435	-	1.435	2.319

In the study of [18], they compared the accuracy of their algorithm at various distances with other existing studies. In our research, we conducted a similar comparison, as presented in Table III, where we compared our results with [18], [17], and [35]. The findings reveal that our proposed algorithm achieves superior accuracy across most distances, except for 100 and 110 centimeters. This comparison underscores the effectiveness of our algorithm in the context of distance measurement compared to other studies.

TABLE III: Comparison between our algorithm and other studies in distance measurement accuracy.

Actual distance (cm)	Accuracy(%)		
	Our algorithm	[18]	[17]
60	**99.57**	99.56	88.00
70	**99.92**	98.95	92.00
80	**99.37**	99.26	98.00
90	**99.87**	99.52	98.00
100	99.67	**99.83**	98.00
110	99.55	**99.67**	99.00
120	**99.46**	98.65	98.50
130	**99.23**	98.21	99.00
140	**99.01**	98.95	99.00
150	**99.97**	99.05	98.00
160	**99.99**	99.26	97.50
170	**99.64**	95.75	97.50
180	**99.324**	95.19	97.00
190	**99.03**	95.19	95.50
200	**98.56**	95.75	95.00

B. Size measurement accuracy

The size measurement aspect of our algorithm encompasses the measurement of width and height for objects of interest, which we will investigate separately for accuracy. Table IV presents the actual width alongside the predicted width and the mean error for each distance in centimeters. The predicted widths closely match the actual widths of objects, and the mean error in centimeters for objects at different distances does not exceed 1 centimeter. These results indicate that our algorithm can accurately measure the width of objects of interest. Additionally, as shown in Table I, the accuracy of our width measurements is 92.87%.

TABLE IV: Error obtained for width measurement without calibration.

distance (cm)	Predicted width(cm)				
	Obstacle 1(10cm)	Obstacle 2(26cm)	Obstacle 3(11cm)	Obstacle 4(7cm)	Mean error(cm)
50	-	26.100	11.578	8.848	0.842
60	9.475	26.039	11.353	8.385	0.576
70	9.377	25.906	11.550	8.517	0.696
80	9.594	26.124	11.696	8.639	0.716
90	9.371	25.712	11.195	8.447	0.640
100	9.568	26.017	11.024	8.377	0.462
110	9.357	26.018	11.245	8.673	0.645
120	9.411	26.054	11.329	8.305	0.569
130	9.363	26.315	11.464	8.149	0.641
140	9.610	25.816	11.414	8.197	0.546
150	9.547	26.047	11.417	8.343	0.565
160	9.731	25.842	11.150	8.434	0.503
170	9.650	25.963	11.116	8.501	0.501
180	9.836	26.031	-	8.393	0.529
190	9.376	25.936	-	8.126	0.605
200	9.244	26.091	-	8.038	0.628

Regarding height measurement, as indicated in Table I, our algorithm achieves an accuracy of 93.92%. In Table V, the highest mean error is observed at a distance of 200 centimeters from the camera, approximately 3 centimeters. Notably, in the study of [14], they reported a similar error level in size measurement, not exceeding 3 centimeters. This fact suggests that our work's height measurement accuracy aligns with comparable studies.

TABLE V: Error obtained for height measurement without calibration.

distance (cm)	Predicted height(cm)				
	Obstacle 1(32.5cm)	Obstacle 2(9cm)	Obstacle 3(9cm)	Obstacle 4(23.5cm)	Mean error(cm)
50	-	9.002	8.913	22.583	0.335
60	27.461	9.004	8.516	21.345	1.920
70	27.983	8.954	8.949	21.404	1.678
80	28.423	9.012	8.986	22.094	1.377
90	28.393	9.029	8.794	21.531	1.578
100	28.481	9.000	8.841	21.472	1.551
110	28.718	9.025	8.934	21.673	1.425
120	29.009	9.029	9.002	21.949	1.268
130	29.246	8.957	9.015	21.602	1.302
140	29.525	8.884	8.828	21.538	1.306
150	29.253	9.061	8.593	21.592	1.406
160	28.762	9.013	8.717	21.673	1.465
170	28.482	8.927	8.485	21.043	1.766
180	27.728	9.125	-	20.698	2.566
190	27.194	9.186	-	20.367	2.875
200	26.745	8.790	-	20.312	3.051

C. Processing time

The average FPS achieved by our implemented algorithm on an Intel Core i5 7th generation CPU is 9 FPS. In the study of [18], which also employed Python and OpenCV for implementation on an Intel Core i7 CPU, their primary focus was distance measurement. Interestingly, their distance measurement accuracy aligns with ours, but they achieved an average processing time of 0.355 seconds, roughly equivalent to 3 FPS. When comparing the FPS of our system to that of the studies of [14] and [2], both of which implemented their algorithms in C++ and achieved 15 and 20 FPS, respectively, a notable disparity in processing time becomes evident. This gap can be attributed to the computational demands of our proposed algorithm, which utilizes YOLO and Neural Networks which requires more system resources.

All the figures and tables presented in this section shows the effectiveness and superiority of the proposed algorithm compared to other alogrithms in the literature. In following section we would conclude the paper.

V. Conclusion

In this paper, we introduced an innovative algorithm that harnesses the capabilities of ML and DL in computer vision to achieve precise distance and size measurements for objects of interest. Our proposed algorithm integrates YOLOv8 for object detection and tracking in combination with template matching within stereo vision systems. The novelty of our study lies in the application of neural networks trained on real-world data to calculate the distances and sizes of detected objects. Furthermore, we demonstrate the algorithm's efficiency through experimentation in two distinct scenarios involving calibrated and uncalibrated cameras resulted in 95.35%, and 94.98% mean accuracy respectively. Even without calibration, our findings demonstrate that our algorithm produces accurate results, often equaling or surpassing other studies in the literature. In the current study, the algorithm achieved the average frame rate of 9 FPS when executed on an Intel Core i5 CPU enough to cover some use cases to be considered real-time. It is worth noting that DL models typically demonstrate superior performance when run on GPUs. Therefore, future research endeavors can prioritize the integration of GPUs to enable real-time implementation of the algorithm in autonomous and robotic systems as a complete subsystem responsible for detection, tracking, and distance and size measurement of objects of interest in the environment.

References

[1] F. Umam, M. Fuad, I. Suwarno, A. Ma'arif, and W. Caesarendra, "Obstacle Avoidance Based on Stereo Vision Navigation System for Omni-directional Robot," *Journal of Robotics and Control (JRC)*, vol. 4, no. 2, pp. 227–242, Apr. 2023.

[2] A. Zaarane, I. Slimani, W. Al Okaishi, I. Atouf, and A. Hamdoun, "Distance measurement system for autonomous vehicles using stereo camera," *Array*, vol. 5, p. 100016, Mar. 2020.

[3] M. Zhou, P. Shen, H. Zhu, and Y. Shen, "In-Water Fish Body-Length Measurement System Based on Stereo Vision," *Sensors*, vol. 23, no. 14, p. 6325, Jul. 2023.

[4] S. Tokoro, "Automotive application systems of a millimeter-wave radar," in *Proceedings of Conference on Intelligent Vehicles*. IEEE, Sep. 1996, pp. 260–265.

[5] F. Nashashibi and M. Devy, "3-D incremental modeling and robot localization in a structured environment using a laser range finder," in *[1993] Proceedings IEEE International Conference on Robotics and Automation*. IEEE, May 1993, pp. 20–27vol.1.

[6] M. Mielle, M. Magnusson, and A. J. Lilienthal, "A comparative analysis of radar and lidar sensing for localization and mapping," in *2019 European Conference on Mobile Robots (ECMR)*. IEEE, Sep. 2019, pp. 1–6.

[7] S. Shirmohammadi and A. Ferrero, "Camera as the instrument: the rising trend of vision based measurement," *IEEE Instrum. Meas. Mag.*, vol. 17, no. 3, pp. 41–47, Jun. 2014.

[8] N. Aswini and V. Uma S, "Obstacle avoidance and distance measurement for unmanned aerial vehicles using monocular vision," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 5, p. 3504, Oct. 2019.

[9] L. Huang, Y. Chen, Z. Fan, and Z. Chen, "Measuring the absolute distance of a front vehicle from an in-car camera based on monocular vision and instance segmentation," *J. Electron. Imaging*, vol. 27, no. 04, p. 1, Jul. 2018.

[10] L. Yang, P. Luo, C. C. Loy, and X. Tang, "A large-scale car dataset for fine-grained categorization and verification," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Jun. 2015, pp. 3973–3981.

[11] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[12] M. N. A. Wahab, N. Sivadev, and K. Sundaraj, "Target distance estimation using monocular vision system for mobile robot," in *2011 IEEE Conference on Open Systems*. IEEE, Sep. 2011, pp. 11–15.

[13] K. A. Rahman, Md. S. Hossain, Md. A.-A. Bhuiyan, T. Zhang, Md. Hasanuzzaman, and H. Ueno, "Person to Camera Distance Measurement Based on Eye-Distance," in *2009 Third International Conference on Multimedia and Ubiquitous Engineering*. IEEE, Jun. 2009, pp. 137–141.

[14] Y. M. Mustafah, R. Noor, H. Hasbi, and A. W. Azma, "Stereo vision images processing for real-time object distance and size measurements," in *2012 International Conference on Computer and Communication Engineering (ICCCE)*. IEEE, Jul. 2012, pp. 659–663.

[15] S.-D. Wei and S.-H. Lai, "Fast template matching based on normalized cross correlation with adaptive multilevel winner update," *IEEE Trans. Image Process.*, vol. 17, no. 11, pp. 2227–2235, Nov. 2008.

[16] I. Slimani, A. Zaarane, and A. Hamdoun, "Convolution algorithm for implementing 2D discrete wavelet transform on the FPGA," in *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*. IEEE, Nov. 2016, pp. 1–3.

[17] F. Martinez, E. Jacinto, and F. Martinez, "Obstacle detection for autonomous systems using stereoscopic images and bacterial behaviour," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 2, pp. 2164–2172, Apr. 2020.

[18] E. Adil, M. Mikou, and A. Mouhsen, "A novel algorithm for distance measurement using stereo camera," *CAAI Trans. Intell. Technol.*, vol. 7, no. 2, pp. 177–186, Jun. 2022.

[19] Z. Wang, Y. Ding, T. Zhang, and X. Huang, "Automatic real-time fire distance, size and power measurement driven by stereo camera and deep learning," *Fire Saf. J.*, vol. 140, p. 103891, Oct. 2023.

[20] D. Brown, "Close-Range Camera Calibration," 1971, [Online; accessed 27. Aug. 2023]. [Online]. Available: https://www.semanticscholar.org/paper/Close-Range-Camera-Calibration-Brown/1150007b62a3c7dac99c2c8f85c63bfab74891af

[21] J. G. Fryer and D. C. Brown, "Lens distortion for close-range photogrammetry," *Photogramm. Eng. Remote Sens.*, vol. 52, pp. 51–58, Jan. 1986. [Online]. Available: https://ui.adsabs.harvard.edu/abs/1986PgERS..52...51F/abstract

[22] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. IEEE, Dec. 2001, vol. 1, p. I.

[23] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. IEEE, Jun. 2005, vol. 1, pp. 886–893vol.1.

[24] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, Jun. 2009, pp. 248–255.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[26] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[27] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Nov. 2013.

[28] R. Girshick, "Fast R-CNN," in *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, pp. 07–13.

[29] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2016.

[30] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[31] N. Aharon, R. Orfaig, and B.-Z. Bobrovsky, "Bot-sort: Robust associations multi-pedestrian tracking," *arXiv preprint arXiv:2206.14651*, 2022.

[32] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, and X. Wang, "Bytetrack: Multi-object tracking by associating every detection box," 2022.

[33] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *Computer Vision – ECCV 2014*. Cham, Switzerland: Springer, 2014, pp. 740–755.

[34] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML'10: Proceedings of the 27th International Conference on International Conference on Machine Learning*. Madison, CT, USA: Omnipress, Jun. 2010, pp. 807–814.

[35] S. Solak and E. D. Bolat, "A new hybrid stereovision-based distance-estimation approach for mobile robot platforms," *Comput. Electr. Eng.*, vol. 67, pp. 672–689, Apr. 2018.