UNIVERSITY OF SCIENCE AND TECHNOLOGY OF HANOI

GROUP PROJECT REPORT

AN INDOOR ASSISTANT FOR BLIND PEOPLE USING OBJECT DETECTION AND DEPTH ESTIMATION

Supervisor: Assoc.Prof.Dr. Nguyen Duc Dung (IOIT)

BI12 – 008	Nguyễn Thiên Ân
BI12 - 019	Vũ Quỳnh Anh
BI12 - 059	Phan Thanh Bình
BI12 - 263	Châu Phan Phương Mai
BI12 - 452	Trần Thế Trung
BA11 - 093	Trần Minh Trung

Academic Years: 2021 - 2024

Hanoi, January 2024

Acknowledgements

We commence by expressing our sincere gratitude to Assoc.Prof.Dr. Nguyen Duc Dung, our supervisor, for his invaluable support, and guidance throughout our project. Despite being extremely busy with his jobs, he spent time hearing, guiding, and keeping us on the correct path. Without his precious assistance and advice, we would never have been able to complete our work. We came to know about so many new things that we are really thankful to him.

Abstraction

Blind people face numerous difficulties in their daily lives, especially in navigating and gaining knowledge of the surrounding environment. Traditional assistive devices such as white canes or guide dogs are limited to range and features while electronic travel aids (ETAs) currently on the market are very expensive and not very portable. The aim of this project is to provide a real-time, low-cost, simple indoor assistant to help visually impaired people avoid collisions and operate more independently on a daily basis. The proposed system consists of three main components: object detection carried out by YOLOv8s, distance estimation based on MiDaS and pyttsx3 library to deliver auditory feedback in real-time. It contains two utilities: warning when close obstacles are present and providing surrounding environment information when the user desires. 11 FPS was achieved for the whole system. We tested the system on three real-world scenarios: classroom, corridor and bedroom. The achieved warning result shows within 1.5 meters, our system detects objects and estimates distance well.

Table of Contents

I. Introduction 1

1. Problem Statements 1

2. Proposed System 2

II. System Components 2

1. Depth Estimation 2

a. Related Work 2

b. MiDaS 3

2. Object Detection 5

a. Related Work 5

b. YOLOv8 5

3. Integrated System 8

III. System Evaluation 13

1. Datasets 13

a. Object Detection 14

b. Depth Estimation 16

2. Performance Evaluation of The System Components 17

a. Depth Estimation 17

b. Object Detection 18

3. Evaluation of The System 20

a. System Setup 20

b. Experimental Setup 21

c. Results and Discussion 22

IV. Conclusion and Future Work 26

V. References 27

LIST OF TABLES

Table 1: Comparison of MiDaS and Other Depth Estimation Models. 18
Table 2: Comparison of Performance Evaluation of YOLOv8s and Other Object Detection Models 19

LIST OF FIGURES

Figure 1: System Design. 2
Figure 2: MiDaS's Architecture. 3
Figure 3: YOLOv8's Architecture 6
Figure 4: Comparison of Focal loss (FL) and Cross Entropy [29]. 7
Figure 5: Result of using trigonometric method 9
Figure 6: Frame Division 10
Figure 7: Depth Focuses. 11
Figure 8: Example of using Dummy Objects in Warning Function. (a) Original Image; (b) Dummy Object; and (c) Depth Result. 12
Figure 9: Example of Calculating Mean for Detected Object. (a) Detected Objects; (b) Objects on Depth Map; and (c) Bounding box for calculation. 12
Figure 10: Examples of our custom dataset. (a), (b) Images from SUN-RGBD; (c) Image from COCO; and (d) Image from the Staircase dataset. 14
Figure 11: Distribution of instances of all classes. 15
Figure 12: Example of Depth dataset. (a) Original Image; and (b) Depth Image. 16
Figure 13: Normalized Confusion Matrix of YOLOv8s. 20
Figure 14: Example of How User Mounts the Camera. 22
Figure 15: Results of our system in a classroom. 23
Figure 16: Results of our system in a bedroom. 23
Figure 17: Results of our system in a corridor. 24
Figure 18. Results of obstacle warning. 25

I. Introduction

1. Problem Statements

The most pervasive sense in our body, vision is essential to every aspect and phase of life. We often take vision for granted, but without it, it is difficult for us to learn, walk, read, engage in school, and work. According to the World Health Organization [1], around 40 million people in the world are blind, while another 250 million have some form of visual impairment. Engaging in day-to-day activities without assistance is a challenging task for a visually impaired person. It becomes more difficult when traveling through unfamiliar locations without a close companion to assist them along the way. Today's assistive devices, such as white canes and guide dogs, can be useful. However, they have some limitations.

The white cane is the most widely used assistive device for blind navigation due to its low cost and simplicity. It is used to detect and avoid obstacles that may be encountered along the way. The cane is moved in an arc about one step ahead of the user. On the other hand, any obstacles outside of this range or above the knee level are not detected. Although white canes are available with advanced technologies, they are costly. Guide dogs are also used in assisting visually impaired persons. A guide dog perceives people as obstacles, so they can move through a crowd without running into anyone, unlike a cane, which requires you to use tactile cues. However, it is not easy to get a trained animal due to the high cost.

Furthermore, a wide range of wearable and portable electronic travel aids (ETAs) have been proposed by numerous studies. Most of these devices use a range of sensors to map their environment and send out audio or visual alerts through headphones. However, many of the ETAs currently on the market are very expensive and not very portable. As a result, offering a blind navigation solution will be extremely beneficial to the blind.

Thanks to the rapid developments of Artificial Intelligence and Machine Learning, more techniques emerged to help visually impaired people navigate. In this project, we developed a visual aid system that relies solely on deep learning models that can identify objects, estimate depth, and provide auditory assistance to help blind individuals navigate in indoor environments, thereby boosting their safety, mobility, and independence. This system can operate in real-time and accept input via a monocular camera, simplifying the design and lowering the cost by reducing the number of sensors necessary.

2. Proposed System

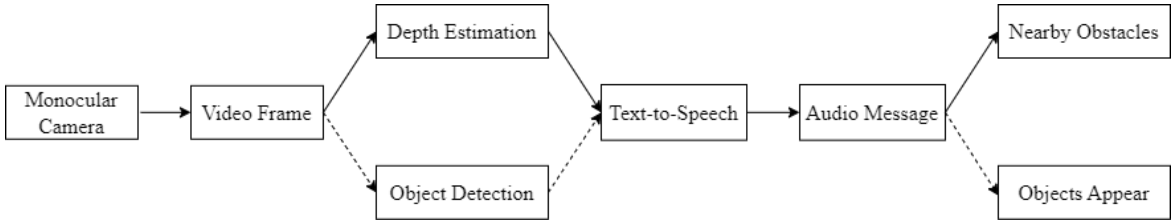


Figure 1: System Design.

Fig. 1 shows the complete workflow of the proposed system. Our system design utilizes a single camera to record, and the video frames will be passed continuously into MiDaS, a monocular depth estimation model. Based on the depth map provided by the model, we can check whether close obstacles are presented in the user's pathway, then alert the user about them to prevent collision and potential accidents. Additionally, users can click a button to engage the object detection module and get a brief description about objects in the frame. We locate the bounding boxes and predict the labels of these objects with the help of a fast and accurate model, YOLOv8. There is one minor condition here: the camera must be held steady to ensure the quality of the image frame so that the model can produce the best results. The depth map in this case is utilized to provide estimated distance information about objects. Both the warning message and the optional description of objects are converted to an audio message spoken to the users using the pyttsx3 module.

In this study, we trained and evaluated our system on a 2020 Intel MacBook Pro with 2GHz Quad-Core Intel Core i5 with 16GB 3733MHz LPDDR4X RAM. The system input was retrieved from this laptop's camera.

II. System Components

1. Depth Estimation
    a. Related Work

In computer vision [2], depth estimation is crucial because it improves perception, enhances reality, and helps comprehend 3D environments. [1]Traditional techniques for image-based depth estimation typically involve triangulating and stereo-matching two 2D pictures taken by binocular cameras to determine their difference to create a depth map. However, binocular depth estimation methods require two or more fixed cameras, and it might be difficult to collect enough matching features in an image when there is little or no texture. Contrarily, monocular depth estimation does not require additional complex equipment or expert procedures but still can produce the depth map from a single camera. To summarize, there are two main

[1] https://www.baeldung.com/cs/disparity-map-stereo-vision

training manners for this approach [2]. The first one is unsupervised learning, which utilizes the abundance of stereo pair-wise images or monocular image sequences over ground truth depth maps and trains the model to minimize the reconstruction loss. On the other hand, supervised learning trains deep neural networks to minimize the pixel-wise discrepancy between its predictions and ground truth depth maps. Some state-of-the-art models are ZoeDepth [3], DDP [4], AdaBins [5].

b. MiDaS

- Overview

One huge problem that exists in depth estimation is that the depth information may be incompatible across different datasets. The reasons behind this might be due to sensor differences, calibration issues, or simply environmental conditions. To handle this, some researchers have proposed a way to solve this problem called Mixed Data Supervision, or MiDaS [6]. This method aims to address the challenges of obtaining accurate depth estimates across diverse datasets.

According to the author of MiDaS, the model is built using the Pytorch framework and has been trained on 10 distinct datasets, including HRWSI [7], TartanAir [8], IRS [9], ReDWeb [10], DIML [11][12][13][14][15], Movies, MegaDepth [16], WSVD [17], ApolloScape [18], and BlendedMVS [19].

After researching to balance between accuracy and inference time, we adopted a small version of MiDaS v2.1. This version is the smallest and fastest of all the MiDaS models in Pytorch, making it suitable for our system's computing resources and performance needs.

- Architecture

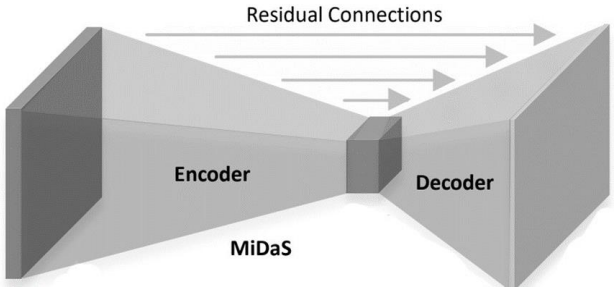


Figure 2: MiDaS's Architecture.

Overall, MiDaS is an encoder-decoder neural network. The encoder part uses EfficientNet [20] which is popular using a technique called scaling (depth, width, and image resolution) to improve a model's performance and efficiency. After successfully extracting features from the image, they will then be fed into a decoder module with multiple Transpose Convolution layers to create our final depth map.

Loss function

The model loss function has two parts. The first part is called scale and shift invariant losses and it can be defined as follows:

$$\mathcal{L}_{ssi}(\hat{d}, \hat{d}^*) = \frac{1}{2M}\sum_{i=1}^{M} \rho\left(\hat{d}_i - \hat{d}_i^*\right)$$

$\hat{d}_i$: $prediction\ at\ pixel\ i^{th}$
$\hat{d}_i^*$: $ground\ truth\ at\ pixel\ i^{th}$
M: $the\ number\ of\ pixels$
ρ: $define\ a\ specifict\ type\ of\ loss\ function$ $(such\ as\ MSE)$

Similar to the scope of this model which is to address ambiguities in the prediction of absolute depth values, this loss formula can solve two parts of scale invariance and shift invariance. The former infers that the loss should not be overly sensitive to the absolute scale of depth predictions meaning that it would not be significantly changed if all predicted values are uniformly scaled. Meanwhile, the latter reduces the change if a constant offset is added to the predicted depth values. All in all, the loss helps the model to be more stable despite the effects coming from absolute depth values or global shifts.

The second part of the loss is a gradient matching term and it simply evaluates the gradient difference between ground truth disparity values and predicted depth at scale k:

$$\mathcal{L}_{reg}(\hat{\mathbf{d}}, \hat{\mathbf{d}}^*) = \frac{1}{M}\sum_{k=1}^{K}\sum_{i=1}^{M}\left(\left|\nabla_x R_i^k\right| + \left|\nabla_y R_i^k\right|\right)$$

M: $the\ number\ of\ pixels$
R_i^k: $difference\ in\ pixels = \hat{d}_i - \hat{d}_i^*\ at\ a\ scale\ k\ feature\ map$

As its name suggests, this term is added to enforce smoothness in the predicted depth map and promotes visually coherent results.

By adding these two formulas we can have the final loss function with a scaling factor alpha.

$$\mathcal{L}_l = \frac{1}{N_l}\sum_{n=1}^{N_l} \mathcal{L}_{ssi}\left(\hat{\mathbf{d}}^n, \left(\hat{\mathbf{d}}^*\right)^n\right) + \alpha\mathcal{L}_{reg}\left(\hat{\mathbf{d}}^n, \left(\hat{\mathbf{d}}^*\right)^n\right)$$

2. Object Detection
    a. Related Work

In computer vision, object detection [21] is one of the quickly emerging fields of technology. An object detection system's principal input is a video frame or picture. The method then locates the observed object using bounding boxes. Each recognized object is assigned a class label, which indicates its category or type.

There are primarily two types of frameworks for generic object detection techniques. One uses the conventional pipeline, first producing region proposals, and then grouping each proposal according to various object categories. The other uses a single framework to obtain outcomes directly, viewing object detection as a regression and classification problem.

Region proposal-based frameworks (Faster R-CNN [22], R-FCN [23]) are composed of several correlated stages, including region proposal generation, feature extraction with CNN, classification, and bounding box regression, which are usually trained separately. Because of this, handling various components takes time, which becomes an issue in real-time applications. One-stage frameworks (YOLO [24], SSD [25]) based on global regression/classification, mapping straightly from image-divided regions to bounding box coordinates and class probabilities, can greatly reduce the time expense.

    b. YOLOv8

- Overview

YOLO is a powerful object detection approach that is widely used in computer vision because of its high processing power. The newest model in the YOLO series is known as YOLOv8, created by [2]Ultralytics, the same firm that built the well-known and industry-defining YOLOv5 [26] model. The reason we chose YOLOv8 was because of its speed and precision. This model is capable of processing images and videos in real-time, even on devices with low resources. Furthermore, YOLOv8 is highly accurate, ensuring reliable object detection results.

YOLOv8 was trained on Microsoft COCO [27], a large-scale image recognition dataset used for object detection, segmentation, and labeling tasks. It comprises more than 330,000 photos, each labeled with 80 object categories and 5 descriptions that describe the scenario.

- Architecture

Most object detection modes' architectures are usually composed of three parts: Backbone, Neck, and Head.

[2] https://docs.ultralytics.com/

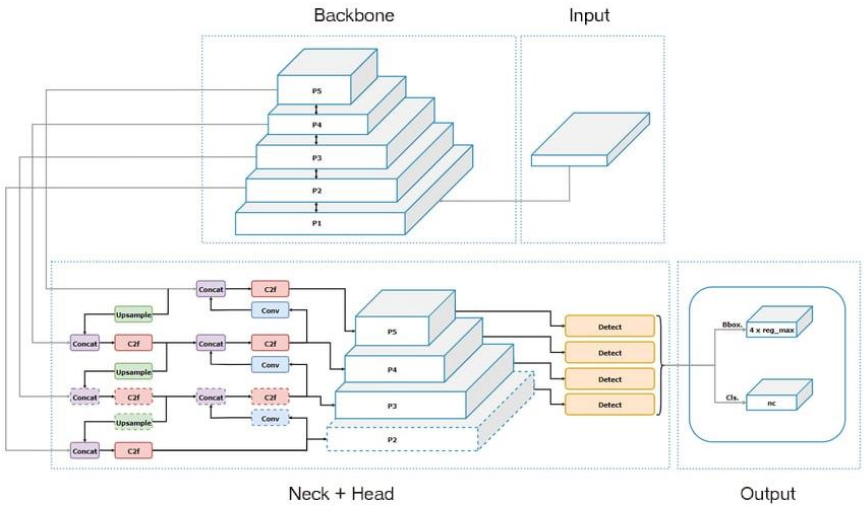


Figure 3: YOLOv8's Architecture.

Backbone

The backbone is the foundational part of the object detection model. It is responsible for extracting features from the input image. These features capture hierarchical representations of the image, ranging from fine-grained details to high-level semantic information.

Apart from some traditional convolution layers, we can also see that YOLOv8 also implements a module called CSPLayer or C2F layer. The C2F module aims to enhance the representational capacity of the network, allowing it to capture and utilize both low-level and high-level features for object detection.

At the end of the backbone, we also have a special layer called Spatial Pyramid Pooling - Fast. The SPP module solves the problem of dealing with objects of different sizes within an image, enables the model to handle objects of various sizes, and also makes the module run twice as fast as the original one.

Neck

Coming after the backbone is the neck. Its primary function is to enhance the representation of features extracted by the backbone. YOLOv8 uses the same neck architecture as YOLOv5 which is Path Aggregation Network (PAN) which solves the problem of leaking information existing in some old methods such as Feature Pyramid Network since now with additional paths, the model has shortened the information path from feature maps which will also enhance feature aggregation across different spatial resolutions.

Head

Moving to the final part which is the final component of an object detection model,

the head. It is responsible for making predictions based on the refined features generated by the backbone and neck.

However, most of the previous YOLO models usually use coupled heads leading to a problem of conflicts between tasks (classification and regression) and this is what makes YOLOv8 different by using a decoupled one.

Another modification from YOLOv8 lies in the prediction technique. Anchor box was being used throughout the predecessors of YOLOv8. Noticing some weaknesses such as the lack of generalization, high complexity of the old methods YOLOv8 uses an anchor-free by simply making a single prediction for each grid cell and directly predicting the offset of an object.

Loss function

There are several loss functions that we can choose from for our model.

- *Classification loss function:*

Focal loss [28]: The traditional object detection problem usually applies cross entropy as the loss function.

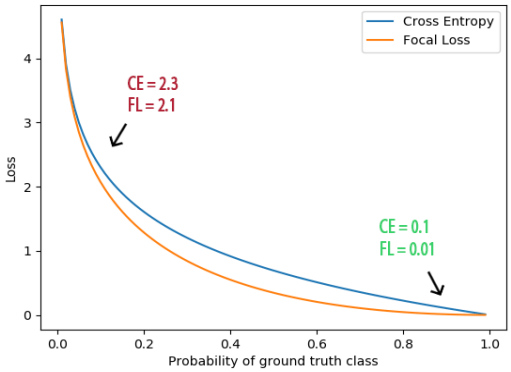


Figure 4: Comparison of Focal loss (FL) and Cross Entropy [29].

However, cross entropy is more sensitive to the imbalance between positive and negative instances which is a headache problem for object detection tasks and focal loss is the one designed to address this issue. Focal loss puts less weight on those with high certainty, and decreases their impact on the loss function, thus, minimizing the data imbalance problem on the loss function. This is done by simply adding a scaling factor as the following formula.

$$\mathrm{FL}(p_t) = -(1 - p_t)^{\gamma} \log(p_t)$$

p_t: *predicted probability for a class*
γ: *focusing parameters*

- *Bounding box loss*

Complete IOU Loss (CIoU): Normal IoU loss function usually only cares about the overlapping area between the ground truth box and the predicted box. However, there are some other criteria that we need to take into consideration and that's what CIoU tries to address.

$$\mathcal{L}_{CIoU} = 1 - IoU + \frac{\rho^2(b, b^{gt})}{c^2} + \alpha v$$

IoU: $Tradition\ Intersection\ over\ Union$
$\rho^2(b, b^{gt})$: $Distance\ between\ two\ centers\ of\ prediction\ and\ ground\ truth\ box$
v: $difference\ in\ aspect\ ratio\ between\ prediction\ and\ ground\ truth$
α: $scaling\ factor$

As you can see, the formula simply adds two more terms considering two more problems of the bounding box which is the distance between centers and the aspect ratios.

- *Combine*

Apart from the two separate loss function types above, there is also an option where the loss function will take into account both the classification loss and the bounding box loss and it is called Varifocal Loss [30]. This loss will be based on a new score called the IOU-aware classification score (IACS) which considers both the IOU of the predicted bounding box and the correctness of the object's class prediction. The formula for this function is as follows:

$$\text{VFL}\ (p, q) = \begin{cases} -q(q\log\ (p) + (1-q)\log\ (1-p)), & q > 0 \\ -\alpha^{\gamma}\log\ (1-p), & q = 0 \end{cases}$$

in which p is the predicted IACS and q is the target value. For the foreground class, q will be assigned as the IOU between the predicted bounding box and the ground truth while it will be 0 for the background.

3. Integrated System

For individuals with visual impairments, delivering information through audio proves to be most beneficial. Given the absence of sight, they depend on heightened sensory perception to engage with and comprehend their surroundings. Hence, we first integrate our results into text and then using text-to-speech (TTS) module, we are able to deliver the information in audio.

- Integrate results to text

Object directional position estimation:

Object directional position is crucial for describing the placement of objects. This helps guide users in finding objects and helps describing the scene in more detail for vision impaired people.

- *Using trigonometric calculation*

There are several ways we could use to do this task, for instance, we could use the center of the frame as the origin and then calculate the position of the objects on a 2D plane using their respective bounding centers.

A way we tried to use is motivated by the exponential representation of a complex number and then calculate the numbers using something like Euler's formula.

$$z = r \cdot e^{i \cdot \varphi} \qquad e^{i \cdot \varphi} = \cos(\varphi) + i \sin(\varphi)$$

Euler's formula is a way to establish the relationship between complex exponential functions and trigonometric functions. With this we can determine the position of the object with respect to the center of the frame based on the trigonometric calculations. The cosine value represents the left-right while the sine value represents the top-bottom position of the object.

For example, we calculated the position of the shelf based on this method.

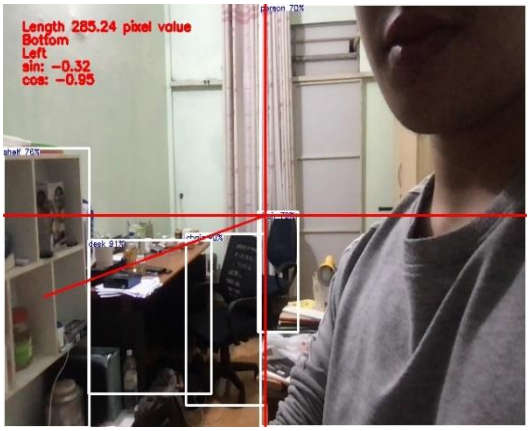


Figure 5: Result of using trigonometric method.

While employing trigonometric calculations to pinpoint an object's position, the resulting numerical output may be difficult to comprehend. To address this issue, we introduce an alternative method involving frame division. This approach aims to provide more efficient information generation compared to the challenging textual nature of trigonometric calculations.

- Using frame division

The first thing we have to do is to divide our frame into 9 equal parts. These parts are used to indicate an exact position of the object with respect to the frame thus providing a strong estimation for the users.

Figure 6: Frame Division.

Objects viewed can be small or large, so it is not a good idea to just base on the object's center to determine the position of the object. With this we use intersection over base area (IOB) to find where the position of the object is.

The intersections over areas are calculated by getting the intersection of the object's bounding box with each of our sub-frames.

$$Intersection\ over\ Area = \frac{Intersection\ Area}{Area\ of\ the\ base\ box}$$

Setting the threshold of 0.5, any objects that pass this threshold are then estimated to be at the respected area.

For large-viewed objects which have a large bounding box and large IOB which could result in multiple corresponding areas, hence to determine the position, we are more biased toward the centers of the frame. The classification priority will be in the following order: "*Mid mid*" > "*Bottom mid*" > "*Top mid*" > "*Left/Right mid*" > "*Left/Right bottom*" > "*Left/Right top*". Both the "*Bottom mid*" subframe and "*Mid mid*" subframe are regarded as in front of the user, so this will generate a text that tells the users that the object is in front of them.

For small-viewed objects which have a small bounding box and small IOB which are below the threshold and do not match any of our sub-frames, we will then use the center of the object to indicate the object's position. In this case, however, all of the middle frames are viewed separately, not like the case for larger-viewed objects.

Similar to the larger-viewed objects, the smaller-viewed objects will adopt the names of the sub-frames and produce text according to their respective locations.

Distance Estimation

Distance estimation is crucial for describing the position of the object as well as providing a warning to the user. There are many ways to get an absolute distance from the depth estimation using stereo camera systems but for a monocular camera system, it is really hard to do so. As such, we provided a way for distance estimation which will result in only two values, far and close.

- Mean Pixels Method

The distance estimation of the warning text utilizes the same method of dividing frames into sub-frames, just like in directional position estimation. Because we want to warn the user if there is something too close in front of them, our focus is on the center sub-frames.

We estimate the distance by calculating the mean of the closest center pixels from the depth output in our sub-frames, additionally since there could be some short-viewed objects that do not span the entire bottom frame, we also calculate the mean of the lower half of the concentrated bottom frame. If either one of the four sub-frames result in a value above the preset threshold, this will provide a warning as something is too close to the user.

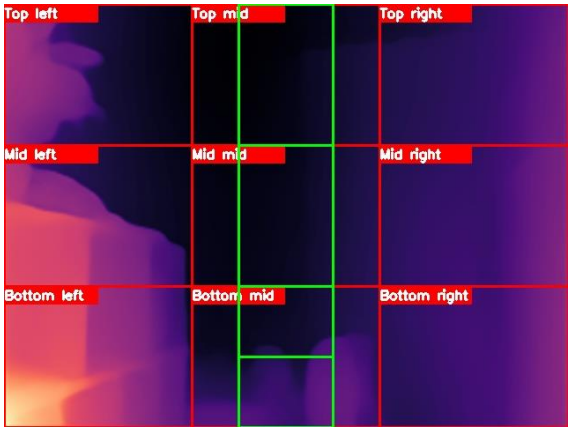
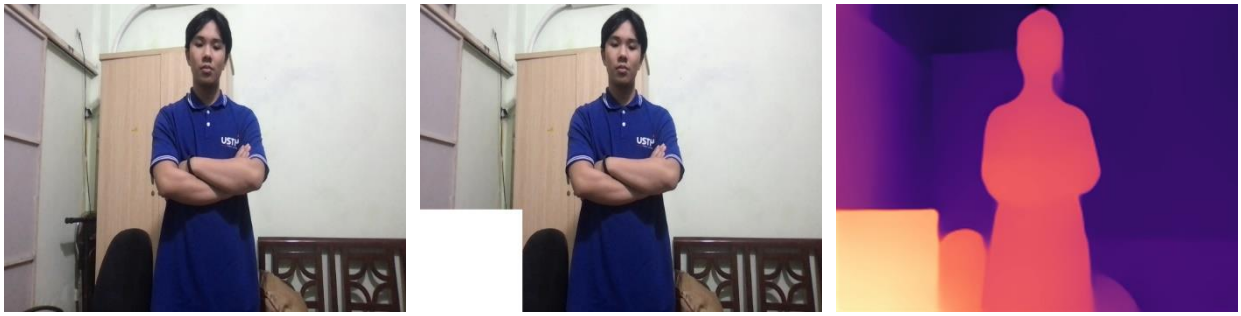


Figure 7: Depth Focuses.

Though this holds some promise in estimating the distance values, we met problems such as the output value from our depth model varies due to varying distribution of pixels and due to the model only generating depth values relatively. To try to somehow fix this problem, we drew a dummy object to the bottom left subframe to use as an anchor object for everything else to scale based on this dummy object. The output of the depth model is somewhat biased towards the bottom, so drawing the

dummy object will result in its being the closest object according to the model. Since we only care about the center sub-frames, the dummy object will not affect our distance estimation for warning. This fixes the problem of depth value varying too much. All there left to do is to set up a threshold based on real world experiments.

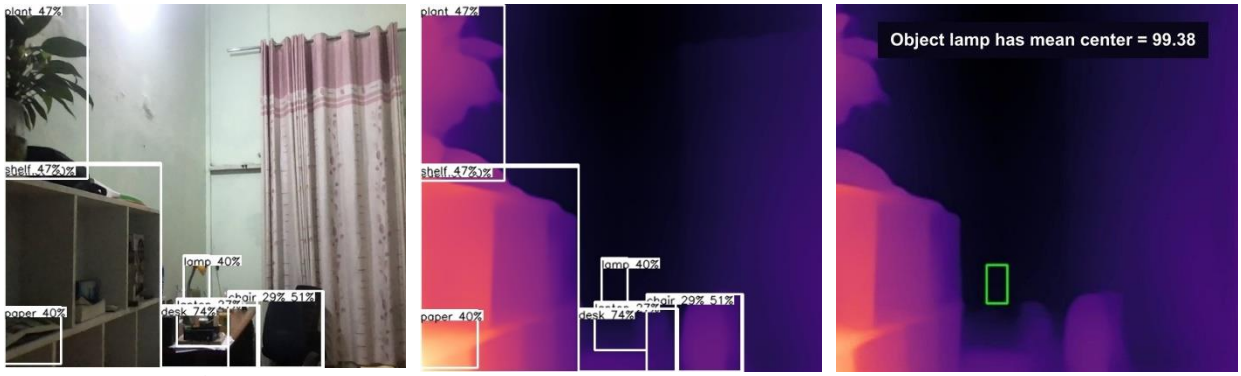


(a) (b) (c)

Figure 8: Example of using Dummy Objects in Warning Function. (a) Original Image; (b) Dummy Object; and (c) Depth Result.

As for the objects, distance estimation is done in a similar way, but without having to draw a dummy base object. For each bounding box, we calculate the mean of the closest object center's pixels from the depth output that covers 50% of the bounding box. The object's depth value is pretty much more evenly distributed due to having bounding boxes for each object, thus it is easier to find the most distributed pixels near the center of each respective object.

The threshold for the object's distance will be different from the threshold for warning since the calculation is done a bit differently.



(a) (b) (c)

Figure 9: Example of Calculating Mean for Detected Object. (a) Detected Objects; (b) Objects on Depth Map; and (c) Bounding box for calculation.

Though this method is easy to understand, this is not an optimized method. In fact, we have not yet figured out an optimized method for distance estimation using a

monocular camera. In theory, it is impossible to figure out the absolute distance only from the values of the depth map.

This method still faces drawbacks such as overlapping bounding boxes which results in ambiguous distance estimation or problems such as uneven distribution of depth pixels. But for now, it is the most simple and efficient way to estimate the distance of the objects.

Final text output
Since our system has two major features, we generate the text output for different features differently.

For the warning feature, the users can choose to turn off the audio if they are already inside their comfort environment since the distance estimation will provide warning if the user is too close to something repeatedly which is pretty annoying. If the user is too close to something in front of them, the text output would be:

"*Object in front of you!*"

This can also be replaced by a "beep" sound.
For each object detected in the object detection and distance estimation, we provide a distance estimation with the two outputs being far and close. We also provide the position of the object relative to the frame and count each object having the same output. So, the output for detected objects may look like this.

"*{number of same class objects} {object's name} {far/close} {position}*"

This is the list of values of the position: *Left*, *right*, *bottom left*, *bottom right*, *top left*, *top right*, *front*, *above (top mid)*, *bottom.*

- Text to Speech module

Based on our system objectives, we simply require a real-time and lightweight module, which Pyttsx3 may meet. Additionally, this library was chosen since it can be utilized in offline applications and is easily deployed consistently across different environments. Moreover, Python also allows converting text to speech in various languages with ease and voice properties such as pitch, speed, and emotion can be customized to suit various needs. The module says 200 words per minute in a man's voice.

III. System Evaluation

1. Datasets

In this project, we trained and evaluated our models using SUN-RGBD [31]. This dataset was chosen because it includes a large number of indoor objects and rooms,

which meets our requirements. It is a benchmark suite designed to improve the state-of-the-art in all major scene interpretation tasks. The Princeton Vision and Robotics Group developed the SUN RGB-D dataset as an expansion of the SUN dataset, which focuses on 2D scene interpretation. The collection includes both RGB photos and depth information. This dataset contains 10,335 RGB-D images taken by four distinct sensors on a scale similar to the PASCAL VOC [32] object detection benchmark.

SUN-RGBD consists of three components:

- RGB Images: High-resolution color images of indoor scenes.
- Depth Maps: Corresponding depth maps of each RGB image.
- Annotations: The dataset is annotated with 2D, 3D bounding boxes for objects, room layout information, and object instance segmentation masks.

a. Object Detection

Using the model that was trained on COCO will not be appropriate because the number of indoor classes in this dataset is insufficient for our goal. As a result, we fine-tuned YOLOv8 to improve its ability to recognize indoor items. This is significant because robust and efficient indoor object detection can assist people with severe vision impairment in autonomously navigating unfamiliar interior environments. We mixed 10335 images of room scenes from the SUN-RGBD dataset with 2000 samples from the COCO dataset and approximately 500 images from the staircase dataset [33].

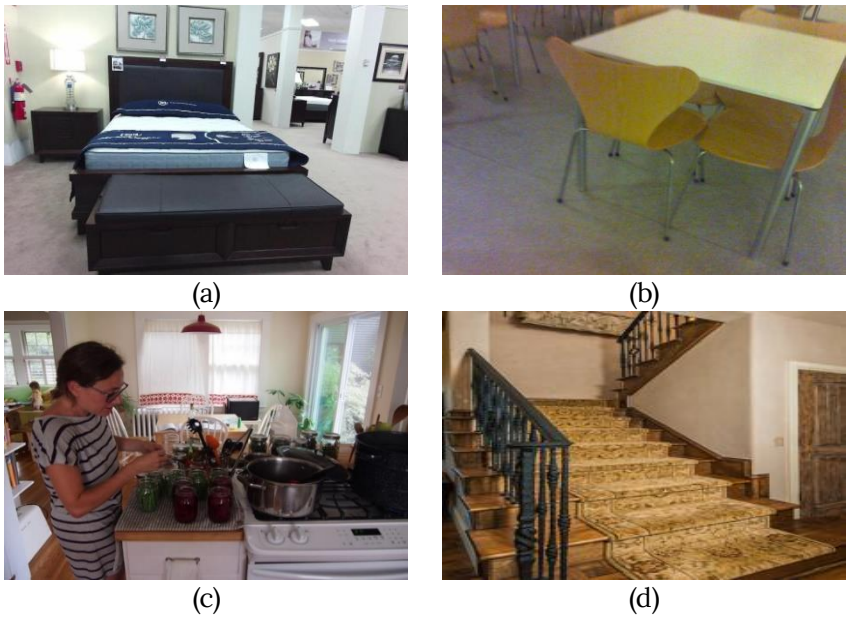


Figure 10: Examples of our custom dataset. (a), (b) Images from SUN-RGBD; (c) Image from COCO; and (d) Image from the Staircase dataset.

Before we can use these three datasets, we must first prepare them. Firstly, for the SUN RGB-D, the bounding boxes have to be recalculated to match the format used in YOLO using these following formula:

$$x_{\text{center}} = (x_{\text{min}} + \text{width}/2)/\text{image width}$$
$$y_{\text{center}} = (y_{\text{min}} + \text{height}/2)/\text{image height}$$
$$\text{width} = \text{width}/\text{image width}$$
$$\text{height} = \text{height} / \text{image height}$$

The dataset originally included 1068 categories relevant to indoor environments. However, the number of objects in a class is not evenly distributed; many classes have fewer than ten objects. As a result, we only kept classes that appear more than 200 times in the dataset, which is only 43 categories.

This is the list of our remaining classes: *bed*, *night_stand*, *ottoman*, *dresser*, *lamp*, *pillow*, *mirror*, *chair*, *sofa*, *monitor*, *cabinet*, *table*, *computer*, *door*, *tv*, *box*, *bottle*, *book*, *laptop*, *shelf*, *plant*, *desk*, *fridge*, *recycle bin*, *garbage bin*, *bench*, *bookshelf*, *printer*, *counter*, *toilet*, *sink*, *towel*, *vanity*, *painting*, *drawer*, *keyboard*, *paper*, *whiteboard*, *picture*, *cpu*, *stool*, *curtain*, *cloth*.

Secondly, for the COCO dataset, we will maintain the class "*person*" and the class that also occurred in the SUN RGB-D, while removing the others. For the bounding boxes of these two datasets, we recalculated using the same formula as in the SUN-RGBD dataset.

- Data Splitting

We will split the dataset into train and validation by randomly shuffling the picture list, ensuring that each class has 80% in the train set and 20% in the validation set.

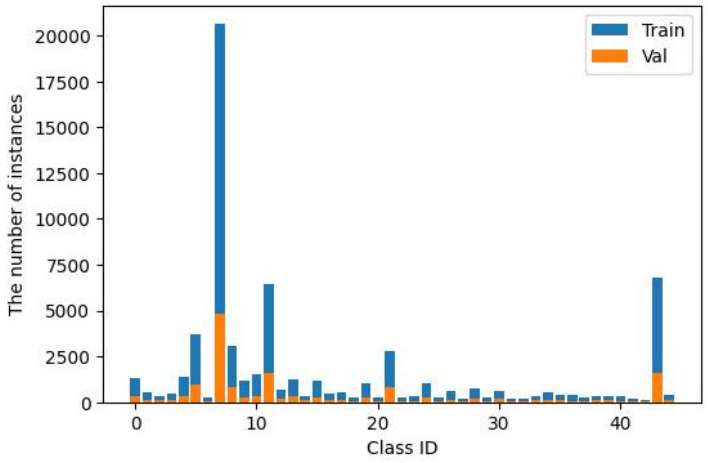


Figure 11: Distribution of instances of all classes.

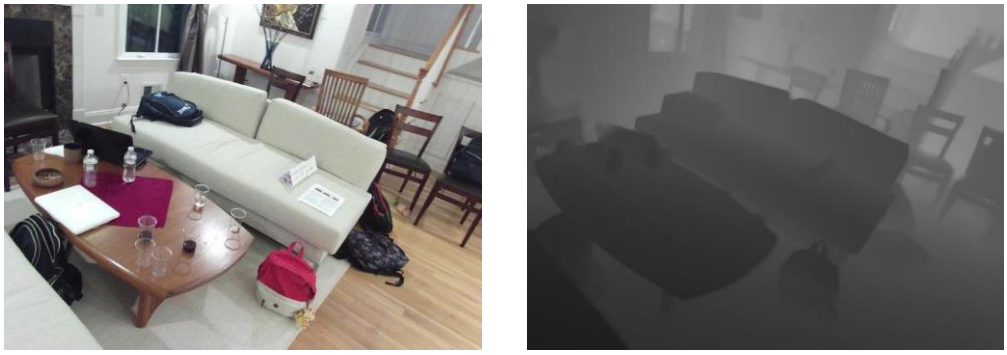
- Data Preprocessing

We use some data augmentation techniques to help improve the model's performance, speed, and accuracy, such as random adjustments to an image's Hue, Saturation, and Value (HSV) channels, or applying a random horizontal or vertical flip to an image with a given probability, randomly rotation and so on. This will increase the diversity of the training data, which can help improve the model's ability to generalize to new data.

Moreover, by dividing each pixel by 255, YOLO rescales the image to a value between 0 and 1. This reduces the ranges of the pixel values, making the model more stable during the training phase.

b. Depth Estimation

RGB images and their depth maps (e.g. *Fig. 12*) are needed for monocular depth estimation tasks. They provide the distance from the camera to each pixel in the images. Note that darker regions represent objects that are closer to the camera and vice versa.



(a) (b)

Figure 12: Example of Depth dataset. (a) Original Image; and (b) Depth Image.

- Data Splitting

For the depth estimation task, the problem of splitting data would be much simpler compared to object detection since we do not have to worry about the distribution of classes. Thus, the data will be simply divided into two parts randomly: train and validation set with the ratio of 8, 2 respectively.

- Data Preprocessing

To prepare data for the training phase, we first apply some augmentation techniques such as random rotation or color jitter. As we have stated above, this would help the model become more generalized to many scenarios and increase its robustness to some outliers. Then, we also rescale the image for training purposes.

Apart from those traditional methods, finetuning MiDaS also requires some additional steps. The model was trained on a scale and shift invariant loss, making its output arbitrary and the prediction would have a large magnitude. Therefore, the loss would be dominated by the overall difference between the predicted depth maps and the ground truth. This problem indicates that we must conduct an alignment process to make the prediction match the original one. For this, we will use the same procedure used in [6], averaging the scale and shift parameters of samples in the training dataset.

2. Performance Evaluation of The System Components
    a. Depth Estimation

For quantity assessment of depth maps, we use three types of metrics: Delta accuracy, Absolute relative difference, and Root Mean Square Error.

Delta accuracy

Delta accuracy refers to the similarity between the predicted depth map and the ground truth one. It is calculated by measuring the ratio of each pixel and checking how many percent of these ratios exceed a certain threshold. Usually, there are three distinct threshold values 1.25, 1.25^2, and 1.25^3representing three metrics δ_1, δ_2, and δ_3, respectively. The general formula is as follows:

$$\text{Delta \_accuracy } (\delta_i) = \max\left(\frac{y}{\hat{y}}, \frac{\hat{y}}{y}\right) = \delta < threshold$$

Absolute relative difference (Abs REL)

In contrast to delta accuracy, Abs REL is defined as the difference between the prediction and the ground truth, or in other words, the lower this metric, the better the performance of the model. It computes the L1 distance between the actual depth map and the predicted one, which is rescaled based on the latter.

$$\text{AbsRel } = \frac{1}{N}\sum_{N} \frac{|D_i^* - D_i|}{D_i}$$

Root Mean Square Error (RMSE)

Similar to Absolute relative difference, RMSE also shows a negative correlation to the result of the model. The formula for this is exactly as its name:

$$RMSE = \sqrt{\frac{1}{n}\sum_{p}^{n} (y - \hat{y})^2}$$

Results

Table 1: Comparison of MiDaS and Other Depth Estimation Models.

	$\delta 1$	$\delta 2$	$\delta 3$	Abs REL	RMSE	FPS
DDP [4]	0.825	0.973	0.994	0.128	0.397	< 1
AdaBins [5]	0.771	0.944	0.983	0.159	0.364	< 1
Hu et al. [34]	0.446	-	-	0.306	0.531	4
FastDepth [35]	0.404	-	-	0.376	0.662	6
Midas - EfficientNet	0.688	0.884	0.948	0.218	0.382	5

For evaluation, we have chosen some of the SOTA models such as DDP or AdaBins and some models designed solely for real-time application on lightweight devices. Overall, it is clear that SOTA models nowadays pay most of their attention to increasing the model performance but not making it practical to run in real-time. This is due to the complex techniques such as diffusion or some heavy architecture like transformers that have restricted these kinds of models from application. Moving on to the three real-time models below, we can witness a significant decline in performance which is not that surprising since most of these only try to integrate some lightweight architecture such as MobileNet or EfficientNet to reduce the number of parameters or operations. However, looking at the whole table, Midas still stands out as the one that can acquire a relatively high speed while still maintaining a comparable result to SOTA models.

b. Object Detection

The model's performance can be evaluated using mAP, FPS and confusion matrix.

mAP@50

[3]Mean Average Precision, or mAP, is a performance metric used for evaluating class prediction of object detection models. This metric ranges from 0 to 100. The higher the number, the better it is. Based on the precision-recall curve, it calculates the Average Precision separately for each class and then averages over all classes:

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i$$

AP_i: the AP of class i
N: $the\ number\ of\ classes$

[3] https://learnopencv.com/mean-average-precision-map-object-detection-model-evaluation-metric/

And, mAP@50 is the mean average precision calculated at an intersection over union (IoU) threshold of 0.50.

Confusion Matrix

[4]A confusion matrix is a table used to visualize and quantify the performance of a classification. [5]The ones used in object detection add one column and one row representing the "*background*", which does not belong to any of the existing classes.

Frame per second (FPS)

The term "*frame per second*" (FPS) describes how quickly a model performs within a video or image sequence by processing and analyzing individual frames.

Results

Table 2: Comparison of Performance Evaluation of YOLOv8s and Other Object Detection Models.

	YOLOv6n [37]	YOLOv6s [37]	YOLOv7 tiny [36]	YOLOv8n	YOLOv8m	YOLOv8s (our)
mAP@50	0.454	0.547	0.489	0.501	0.577	0.554
FPS	16	7	10	14	4	8

Table 2 compares our model's performance to other fine-tuned models on the custom dataset. We can see that all of the YOLOv8 variations have higher accuracy than their counterparts. This might be due to the fact that YOLOv8 uses anchor free method meaning that the model may be less complex compared to the predecessors. Taking a closer look into YOLOv8 models, YOLOv8m has the best accuracy, but its FPS is slightly slow; YOLOv8n has a slightly higher FPS but insufficient accuracy for usage in our system. To balance accuracy and quickness, YOLOv8s was the best option.

[4] https://www.datature.io/blog/how-to-evaluate-computer-vision-models-with-confusion-matrix
[5] https://medium.com/@a0922/confusion-matrix-of-mĩv8-97fd7ff0074e

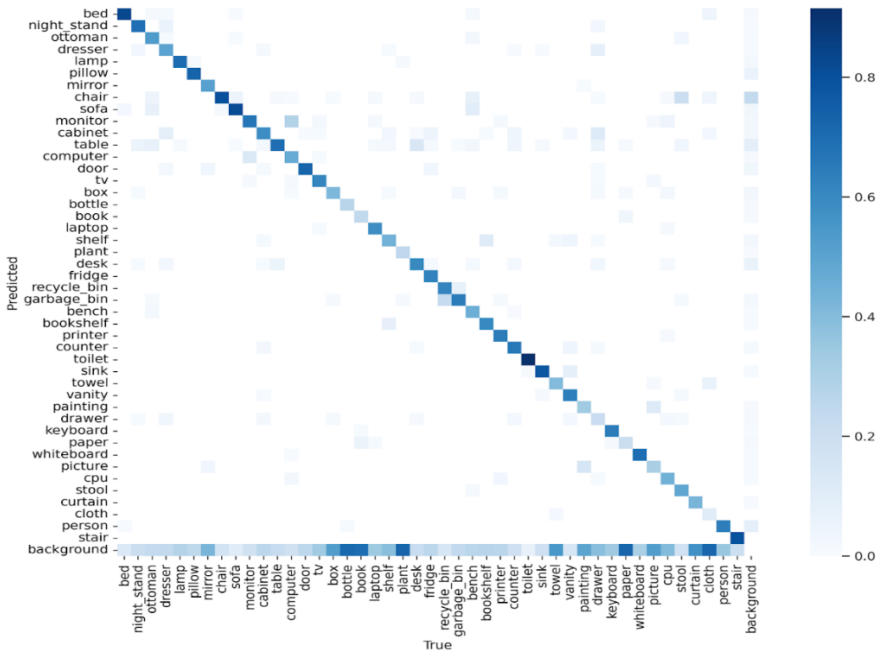


Figure 13: Normalized Confusion Matrix of YOLOv8s.

The above figure is our model's normalized confusion matrix. Our model does a fairly good job of class detection overall. The classes with the highest scores, larger than 0.8, are *bed*, *chair*, *toilet*, *door*, *stair*, *sofa*, *and so on*. The scores for the *person*, *door*, *and box* were also good, ranging from 0.6 to 0.8. But a lot of classes, including *clothing*, *laptops*, *etc.*, only receive a score of 0.0 to 0.2, and many others are mislabeled as *backgrounds*. This might be the result of our dataset's imbalance. Many classes have more than 1,000 objects; especially, the chair class has almost 20,000 instances, which is much higher than the class with the fewest instances (200). Furthermore, the dataset contains objects set against cluttered or intricate backdrops, making it difficult for the model to distinguish between them. Our model also has difficulty in recognizing small objects *such as books or bottles*.

3. Evaluation of The System
    a. System Setup

Preprocess

To preprocess the input before the inference stage, we need to first resize the image to 640x480 resolution. This is to make a compatible size for our two models to run properly. Next, similar to what we have done in the training phase, the image also needs to be rescaled to [0,1] range.

Models

The models will be integrated to generate text as we have discussed above. Also, as we can see from the evaluation section, our models still run at a relatively slow speed, thus, for boosting purposes, we converted them into ONNX format.

ONNX, which stands for Open Neural Network Exchange, is an open format designed to represent deep learning models and allow AI developers to utilize models with a range of frameworks, tools, runtimes, and compilers. Above all, ONNX can also be used to optimize and accelerate speed. In our cases, ONNX models have been shown to achieve a speed nearly 2.5 times higher than the original one used in the Torch package.

Threshold setup for warning and object distance estimation

From our experiments we tried various scenarios where objects are in various places. At the end of our tests, we set the threshold of 650 for the warning feature and 400 for object distance estimation. Each of them corresponds to the estimation of 1.5 meters or each respective feature. This passed our criterias. Since the output of our model is an inverse depth map, thus, an object will be considered as close if the calculated value exceeds the threshold, and vice versa.

Postprocess

After we have gotten the results from the system, the last step is to visualize all of them. To do this, we represent them as four windows as follows:

- The first window displays the original RGB frame that we have taken from the camera.
- The second window is the depth map generated by the MiDaS model.
- The third window will pop up when the user requests, this window combines the result coming from the object detection model which includes bounding box, class and confidence score for each object in the frame.
- The last window shows the text which will be spoken to the user given a scenario.

b. Experimental Setup

The usability and performance of our whole system are tested in three different indoor scenarios: classroom, bedroom, and corridor. We set up the objects so that the system evaluation would yield objective conclusions. For simplicity, the monocular camera would be mounted on the user's chest. In these cases, the user moved forward while the objects were stationary. In order to assess the system's performance, we must have some evaluation criteria.

Figure 14: Example of How User Mounts the Camera.

To simulate that the monocular camera is mounted on the user's chest using laptop camera and phone camera, we simply hold these devices to our chest to record test case videos.

Evaluation criteria

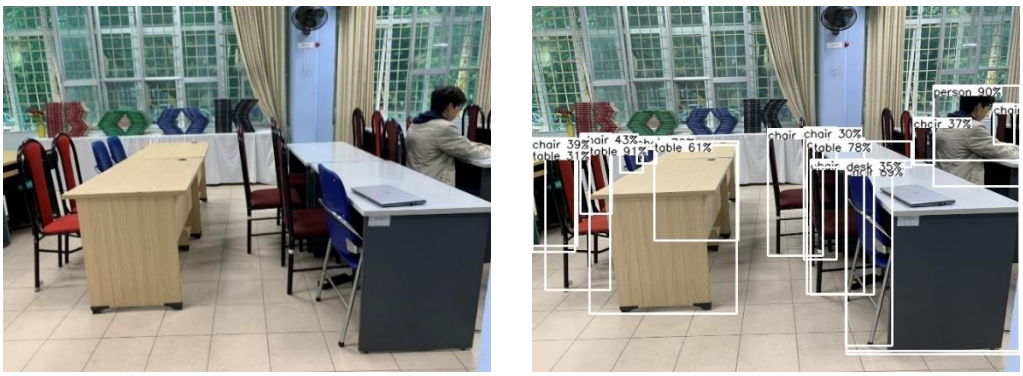
In order to assess the system's performance, we must have some evaluation criteria:

- The system must be fast enough to provide warnings to the user. Since we are only testing on a CPU a speed of 10 FPS would be considered a success.
- The objects detected must be correctly classified into different positions relative to the user.
- The distance estimation for warnings must be far enough so that the users can avoid collision in time. We say that any obstacles that are closer than 1.5 meters in front of the user will trigger a warning.
- The distance estimation for objects must be relevant based on real world scenarios. Therefore, for simplicity, we say that any objects that are in the 1.5 meters range of the user should return a close value and vice versa.

c. Results and Discussion

- Results for object distance estimation

Classroom



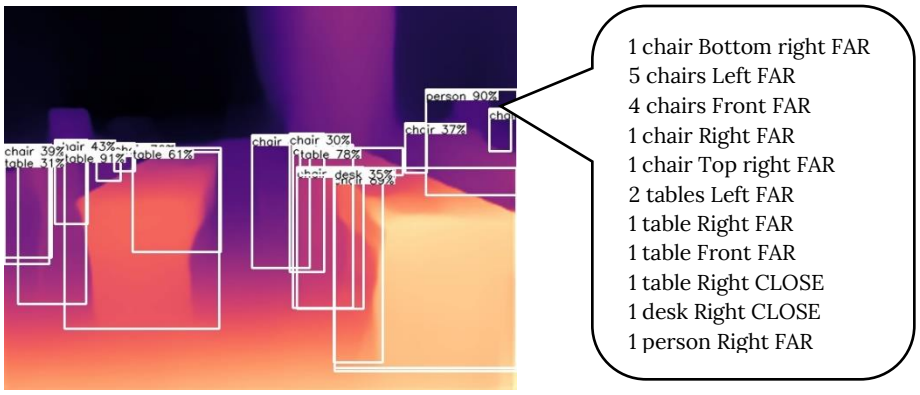


Figure 15: Results of our system in a classroom.

Overall, the object detection model performance is acceptable but still affected by overlapping objects. We can see that some objects are redundantly detected (*a chair*, *a table*) and this consequently affects the final result. Additionally, there are some objects that are not detected because of their small sizes or unclear visibility.

The depth map, on the other hand, shows a clear bias towards large or bright objects. Some relatively small or darker objects such as chairs are not clearly shown on the depth map and it only produces a correct result thanks to the nearby table.

Bedroom

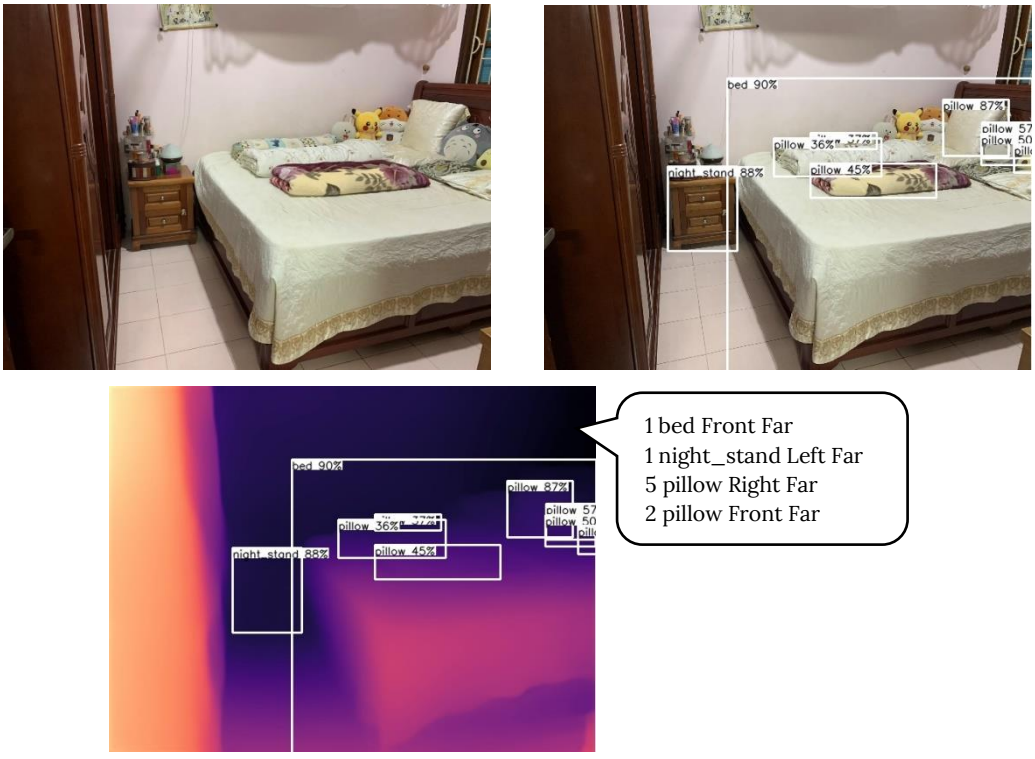


Figure 16: Results of our system in a bedroom.

For this test scenario, one problem is that there are many small-size objects and unsurprisingly, most of them are not recognized by YOLO. Another noticeable error occurring in this case is a miss detection of a huge cabinet. This indicates that our model is not only affected by the size but it is also highly dependent on the angle of objects being viewed.

Meanwhile, the depth estimation model can not seem to be able to detect objects with small sizes or if the color is not distinguished from the background/other objects.

Corridor

Figure 17: Results of our system in a corridor.

For the last test case, both the object detection model and depth estimation seem to perform perfectly. Thus, given a scenario where everything is separate from each other, the performance from YOLO would be exceptional.

However, this does not mean that the final result is good. As you can see, the door is detected at the top mid position which is not correct. This is a problem when we use a fixed grid for the position detection of our system since it is not relative to the range of the image. Hence, for a bigger space like this, we would need to make some modifications to the grid or implement other algorithms that are more flexible than this one.

- Results for obstacle warning

For the obstacle warning we tried with both tall-viewed objects where their pixel distribution would spread around our center subframes and short-viewed objects where their pixels would spread only in the bottom half of our "*bottom mid*" sub-frame.

Figure 18. Results of obstacle warning.

You can see the differences between the two types of obstacles that we ran into. While the shorter viewed obstacles were pretty close, around 1.5 meters of the camera, all of their mean values are smaller than the preset threshold that we tried to develop for our warning feature except the addition calculation of half of the bottom frame mean. This showed the effectiveness of the additional frame that we added on.

- FPS

While running these experiments on the PyTorch framework, we only got an FPS ranging from 4-5 FPS. However, with ONNX, this improved by nearly 2.5 times to around 10-11 FPS which barely passed our preset criteria.

Summary

With regard to the distance estimation function, it is clear that in the indoor environment, our system easily detects large objects like *tables*, *chairs*, *fridges*, *etc.* with high confidence scores. Nevertheless, several objects under a specific condition are hard to identify. Meanwhile, the warning feature works great in various scenarios but it still lacks the ability to warn our users against uniformed scenarios such as walls. Regarding the limitations, the difficulty of depth measurement with only one camera meant that the findings were not precise. It is expected that there is a fair degree of inaccuracy in our final assessment given that there were some mistakes in both the depth perception and camera calibration steps.

IV. Conclusion and Future Work

This work presents the results of the initial efforts to develop a system based on computer vision that has a low cost and supports blind people in their safe navigation. Overall, our system can identify a variety of indoor objects and determine the distance between them and users. The designed system will provide a message if the users desire to learn more about their surroundings or if there are nearby obstacles. However, our system also faces some challenges. These are some of the possible future work directions that can help to solve these problems:

Firstly, there is still no clear connection between our depth result and the real-world distance, thus, finding a way to calibrate would be a huge breakthrough. The camera calibration provides the necessary parameters to convert pixel measurements to real-world coordinates, enabling an accurate determination of the absolute distance. Additionally, 3D reconstruction is also an effective technique that provides accurate depth information for each point in a scene, allowing for precise measurement of distances between objects and the camera.

Secondly, we want to deploy the system into a mobile device so that users may carry it around conveniently.

Finally, to improve the accuracy of the object detection models, we attempt to balance the distribution of classes in our dataset by including more object instances. We can also add more classes to help our system recognize a broader range of interior objects. Furthermore, with the advanced Deep Learning algorithms, we may design and test our system in a more complicated outdoor environment.

V. References

[1] "Blindness and vision impairment," World Health Organization.

[2] Y. Ming, X. Meng, C. Fan, and H. Yu, "Deep learning for monocular depth estimation: A review," *Neurocomputing*, vol. 438, pp. 14–33, 2021, doi: https://doi.org/10.1016/j.neucom.2020.12.089.

[3] S. Bhat, R. Birkl, D. Wofk, P. Wonka, and M. Muller, "ZoeDepth: Zero-shot Transfer by Combining Relative and Metric Depth," *ArXiv*, vol. abs/2302.12288, 2023, [Online]. Available: https://api.semanticscholar.org/CorpusID:257205739

[4] Y. Ji *et al.*, "DDP: Diffusion Model for Dense Visual Prediction," *ArXiv*, vol. abs/2303.17559, 2023, [Online]. Available: https://api.semanticscholar.org/CorpusID:257833921

[5] S. F. Bhat, I. Alhashim, and P. Wonka, "AdaBins: Depth Estimation using Adaptive Bins," *CoRR*, vol. abs/2011.14141, 2020, [Online]. Available: https://arxiv.org/abs/2011.14141

[6] K. Lasinger, R. Ranftl, K. Schindler, and V. Koltun, "Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-Shot Cross-Dataset Transfer," *CoRR*, vol. abs/1907.01341, 2019, [Online]. Available: http://arxiv.org/abs/1907.01341

[7] K. Xian, J. Zhang, O. Wang, L. Mai, Z. Lin, and Z. Cao, "Structure-Guided Ranking Loss for Single Image Depth Prediction," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 608–617. doi: 10.1109/CVPR42600.2020.00069.

[8] W. Wang *et al.*, "TartanAir: A Dataset to Push the Limits of Visual SLAM," *CoRR*, vol. abs/2003.14338, 2020, [Online]. Available: https://arxiv.org/abs/2003.14338

[9] Q. Wang, S. Zheng, Q. Yan, F. Deng, K. Zhao, and X. Chu, "IRS: A Large Synthetic Indoor Robotics Stereo Dataset for Disparity and Surface Normal Estimation," *CoRR*, vol. abs/1912.09678, 2019, [Online]. Available: http://arxiv.org/abs/1912.09678

[10] K. Xian *et al.*, "Monocular Relative Depth Perception with Web Stereo Data Supervision," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 311–320. doi: 10.1109/CVPR.2018.00040.

[11] J. Cho, D. Min, Y. Kim, and K. Sohn, "DIML/CVL RGB-D Dataset: 2M RGB-D Images of Natural Indoor and Outdoor Scenes," *CoRR*, vol. abs/2110.11590, 2021, [Online]. Available: https://arxiv.org/abs/2110.11590

[12] Y. Kim, B. Ham, C. Oh, and K. Sohn, "Structure Selective Depth Superresolution for RGB-D Cameras," *IEEE Transactions on Image Processing*, vol. 25, no. 11, pp. 5227–5238, 2016, doi: 10.1109/TIP.2016.2601262.

[13] S. Kim, D. Min, B. Ham, S. Kim, and K. Sohn, "Deep stereo confidence prediction for depth estimation," in *2017 IEEE International Conference on Image Processing (ICIP)*, 2017, pp. 992–996. doi: 10.1109/ICIP.2017.8296430.

[14] Y. Kim, H. Jung, D. Min, and K. Sohn, "Deep Monocular Depth Estimation via Integration of Global and Local Predictions," *IEEE Transactions on Image Processing*, vol. 27, no. 8, pp. 4131–4144, 2018, doi: 10.1109/TIP.2018.2836318.

[15] J. Cho, D. Min, Y. Kim, and K. Sohn, "Deep monocular depth estimation leveraging a large-scale outdoor stereo dataset," *Expert Syst Appl*, vol. 178, p. 114877, 2021, doi: https://doi.org/10.1016/j.eswa.2021.114877.

[16] Z. Li and N. Snavely, "MegaDepth: Learning Single-View Depth Prediction from Internet Photos," *CoRR*, vol. abs/1804.00607, 2018, [Online]. Available: http://arxiv.org/abs/1804.00607

[17] C. Wang, S. Lucey, F. Perazzi, and O. Wang, "Web Stereo Video Supervision for Depth Prediction from Dynamic Scenes," *CoRR*, vol. abs/1904.11112, 2019, [Online]. Available: http://arxiv.org/abs/1904.11112

[18] X. Huang *et al.*, "The ApolloScape Dataset for Autonomous Driving," *CoRR*, vol. abs/1803.06184, 2018, [Online]. Available: http://arxiv.org/abs/1803.06184

[19] Y. Yao *et al.*, "BlendedMVS: A Large-scale Dataset for Generalized Multi-view Stereo Networks," *CoRR*, vol. abs/1911.10127, 2019, [Online]. Available: http://arxiv.org/abs/1911.10127

[20] M. Tan and Q. V Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," *CoRR*, vol. abs/1905.11946, 2019, [Online]. Available: http://arxiv.org/abs/1905.11946

[21] H. Shenai, J. Gala, K. Kekre, P. Chitale, and R. Karani, "Chapter 4 - Combating COVID-19 using object detection techniques for next-generation autonomous systems," in *Cyber-Physical Systems*, R. C. Poonia, B. Agarwal, S. Kumar, M. S. Khan, G. Marques, and J. Nayak, Eds., Academic Press, 2022, pp. 55–73. doi: https://doi.org/10.1016/B978-0-12-824557-6.00007-8.

[22] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *CoRR*, vol. abs/1506.01497, 2015, [Online]. Available: http://arxiv.org/abs/1506.01497

[23] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object Detection via Region-based Fully Convolutional Networks," *CoRR*, vol. abs/1605.06409, 2016, [Online]. Available: http://arxiv.org/abs/1605.06409

[24] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *CoRR*, vol. abs/1506.02640, 2015, [Online]. Available: http://arxiv.org/abs/1506.02640

[25] W. Liu *et al.*, "SSD: Single Shot MultiBox Detector," *CoRR*, vol. abs/1512.02325, 2015, [Online]. Available: http://arxiv.org/abs/1512.02325

[26] H. N. Noura, O. Salman, R. Couturier, and A. Sider, "A Deep Learning Object Detection Method for an Efficient Clusters Initialization," *CoRR*, vol. abs/2104.13634, 2021, [Online]. Available: https://arxiv.org/abs/2104.13634

[27] T.-Y. Lin *et al.*, “Microsoft COCO: Common Objects in Context,” *CoRR*, vol. abs/1405.0312, 2014, [Online]. Available: http://arxiv.org/abs/1405.0312

[28] T.-Y. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, “Focal Loss for Dense Object Detection,” *CoRR*, vol. abs/1708.02002, 2017, [Online]. Available: http://arxiv.org/abs/1708.02002

[29] S. Kumar, S. Haresh, B. Baloch, A. Rehman, and T. Syed, “Focused-Anchors Loss for imbalanced classification.” Jan. 2018.

[30] H. Zhang, Y. Wang, F. Dayoub, and N. Sünderhauf, “VarifocalNet: An IoU-aware Dense Object Detector,” *CoRR*, vol. abs/2008.13367, 2020, [Online]. Available: https://arxiv.org/abs/2008.13367

[31] S. Song, S. P. Lichtenberg, and J. Xiao, “SUN RGB-D: A RGB-D scene understanding benchmark suite,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 567–576. doi: 10.1109/CVPR.2015.7298655.

[32] “International Journal of Computer Vision manuscript No. (will be inserted by the editor) The PASCAL Visual Object Classes (VOC) Challenge,” [Online]. Available: https://api.semanticscholar.org/CorpusID:4246903

[33] A. Gaikwad, V. Gohokar, and R. Kute, “Staircase Dataset.” Mendeley Data, Jan. 19, 2023.

[34] J. Hu, C. Fan, H. Jiang, X. Guo, X. Lu, and T. L. Lam, “Boosting Light-Weight Depth Estimation Via Knowledge Distillation,” *CoRR*, vol. abs/2105.06143, 2021, [Online]. Available: https://arxiv.org/abs/2105.06143

[35] D. Wofk, F. Ma, T.-J. Yang, S. Karaman, and V. Sze, “FastDepth: Fast Monocular Depth Estimation on Embedded Systems,” *CoRR*, vol. abs/1903.03273, 2019, [Online]. Available: http://arxiv.org/abs/1903.03273

[36] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, “YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors,” *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7464–7475, 2022, [Online]. Available: https://api.semanticscholar.org/CorpusID:250311206

[37] C. Li *et al.*, “YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications,” *ArXiv*, vol. abs/2209.02976, 2022, [Online]. Available: https://api.semanticscholar.org/CorpusID:252110986