

Objective

To determine if off the shelf hardware and object detection techniques can be used to provide live object avoidance to the visually impaired.

Background

Object detection: A computer vision technique that can be used to identify and locate objects in an image or video.

- A computer must classify and localize objects in an image in order to detect those objects
- An object detection model is an algorithm that can classify objects in an image, most commonly using neural networks
- Object detection models can be created and trained to fit certain use cases
- Pre-trained models have been trained according to the needs of previous use cases and can be applied to use cases in similar domains

Object Detection is currently used in various products to aid the visually impaired, such as:

- Biped Smart Harness: an AI copilot for persons who are blind or have a visual impairment
- QR codes are being used at JR Shin-Kobe Station to assist the blind navigate using their smartphone



Methodological Research

- Investigated building a custom object detection model versus using open source pre-trained models

Creating and Training a Model

Pros	Cons
Control of dataset	Large amount of data needed
Can specify for use case	Time needed to train and test
Ability to modify algorithm	Model must be retrained for use

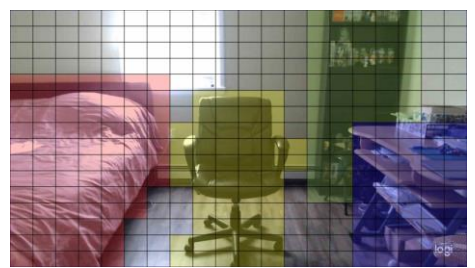
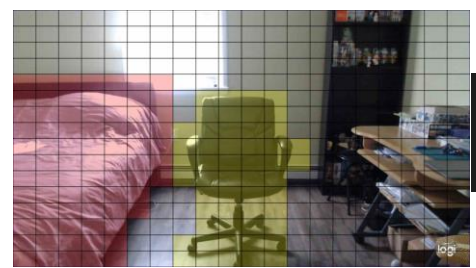
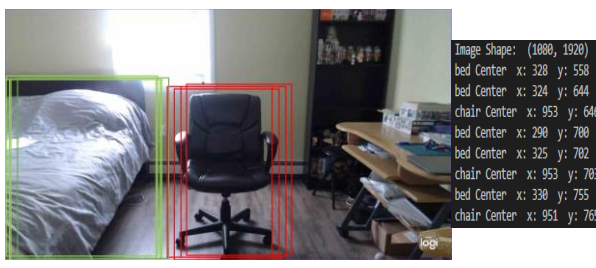
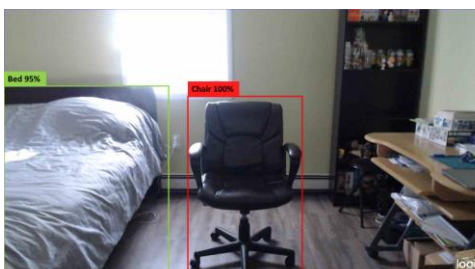

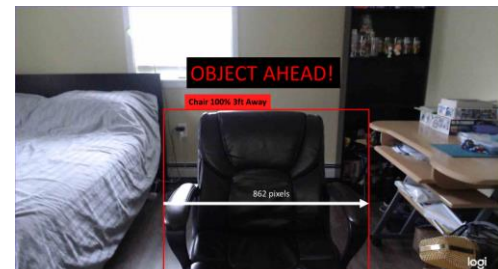
Using a Pre-trained Model

Pros	Cons
Already trained (plug and play)	Cannot specify dataset
Widely used for a general use case	Cannot modify algorithm
Saves time and resources	

- Based on empirical analysis, a pre-trained model was better suited this project
 - Pre-trained open-source models exist to detect everyday household items
 - Training a model would be impractical due to data and computing resource requirements
- The pre-trained models You Only Look Once (YOLO)v3 and Single Shot Detectors (SSD) MobileNet were tested.
- YOLO was more accurate and faster to detect objects in a live video feed than SSD MobileNet, based on testing

Object Detection and Avoidance Algorithm

- Object avoidance systems must detect objects and calculate their distance from the user
- Developed a six-step algorithm to identify the distance to known objects in a live video stream

1. Process frame to detect objects using YOLOv3 weight files	2. Find known objects in object data base	3. Estimate location of object in frame
		
4. Create single bounding box using Non-Maximal Suppression and label object name and confidence	5. If a known measured object is detected, compute distance from camera	6. Determine if user is within distance threshold and trigger an alert if they are
		

- Using a reference image (Step 5) with a constant distance and size, the system calculates the following:

$$\text{Focal Length} = \frac{(\text{widthInPixels} * \text{measuredDistance})}{\text{actualWidth}} \quad \text{Current Distance} = \frac{(\text{actualWidth} * \text{focalLength})}{\text{widthInPixels}}$$

- Calculations for Step 5 and 6:

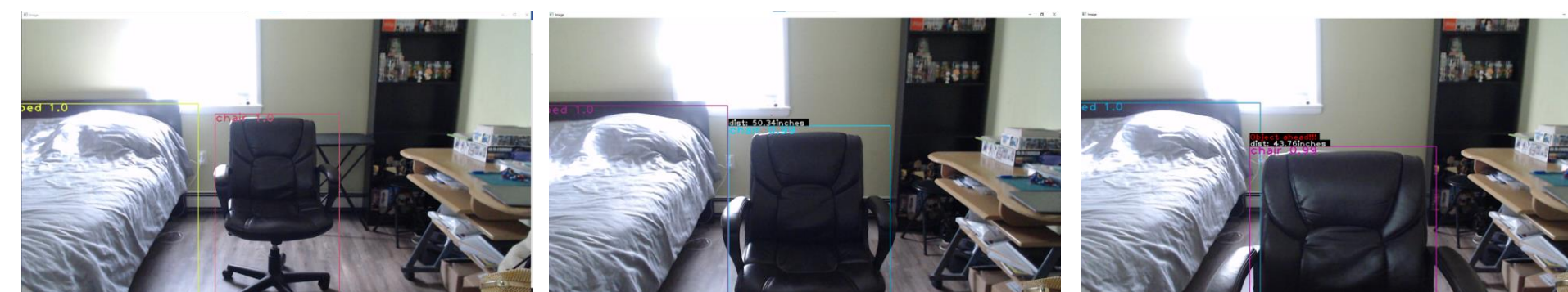
$$\text{Focal Length} = \frac{431 \text{ pixels} * 72 \text{ inches}}{24 \text{ inches}} = 1293 \text{ pixels} \quad \text{Current Distance} = \frac{24 \text{ inches} * 1293 \text{ pixels}}{862 \text{ pixels}} = 36 \text{ inches}$$

Implementation



- Using Python programming language, OpenCV (a Computer Vision library) captures frames in a video feed
- Once captured, each frame is processed using the pre-trained YOLOv3 algorithm to detect objects. Only objects with a confidence score greater than 60% will display a bounding box.
- Known objects will have their current distance to the camera calculated
- If known objects come within a threshold of 48 inches (4ft), an alert is triggered indicating an object is ahead

- Camera: Logitech C922 Webcam (1080p)
- CPU: AMD Ryzen 5 3600
- Graphics Card: GTX 1660 Ti
- Operating System: Windows 10
- IDE: Visual Studio Code
- Versioning: GitHub



Findings and Limitations

An off the shelf, pre-trained object detection model can be used to build an object avoidance system.

The system can:

- Process live video footage and detect objects
- Calculate the size of and distance to known objects
- Trigger an alert to avoid objects once a certain distance threshold is met

Testing and evaluation of the system uncovered several issues that affect the outcome of the object avoidance process.

Limitations include:

- Use of only one camera, restricting depth and spatial capture of footage
- Image quality due to camera resolution, angles, and lighting
- Dependent on accuracy of YOLOv3 algorithm and model

Future Work

- Determine an object's position in 3D space using techniques such as triangulation
- Investigate more specialized hardware such as stereo vision cameras and lidar devices
- Expand number of objects in database that the system can detect and calculate distance to
- Deeper quantitative evaluation to test the benefits of training a custom model
- Conduct tests of the system with users (people with visual impairments)

Conclusions

- Developed a collision avoidance algorithm and system using a pre-trained object detection model
- Objects can be detected and their distance from the camera can be computed
- Objects need to be more consistently and accurately detected by the system before performing user tests
- Further evaluation is required to assess the utility of the system when implemented in the real world

Acknowledgements

This work was supported by the New Jersey Space Grant Consortium and Brookdale Community College. I would like to thank Dr. Christopher Ochs, Professor Karina Ochs, and Professor Ana Teodorescu for their continued assistance and support throughout this project.