“Anecdote: Where Sound Meets Insight”

A Project Thesis

Submitted for the partial fulfillment for the award of the degree of

(Bachelor of Technology 6th Semester, CSE)

Submitted by

1.	2216443	Vibhuti Bhardwaj	CSE
2.	2216446	Vikhyati Singh	CSE
3.	2216319	Mansi Mishra	CSE
4.	2216321	Manya Bhardwaj	CSE

Under the supervision of

Dr. Khandakar F. Rahman
Assistant professor

Department of Computer Science

Banasthali Vidyapith

Certificate

Certified that following students have carried out the project work **titled "Anecdote: Where sound meets insight" from 1st August 2024 to 31st March 2025** for the award of the **Bachelor of Technology** from **Banasthali Vidyapith** under my supervision. The thesis/report embodies result of original work and studies carried out by students themselves and the contents of the thesis/report do not form the basis for the award of any other degree to the candidates or to anybody else.

1.	2216443	Vibhuti Bhardwaj	CSE
2.	2216446	Vikhyati Singh	CSE
3.	2216319	Mansi Mishra	CSE
4.	2216321	Manya Bhardwaj	CSE

Dr. Khandakar F. Rahman

Assistant professor

Place: Banasthali Vidyapith.

Date: 5/4/2025

Abstract

Object detection is a fundamental computer vision technique that involves identifying and localizing multiple objects within an image or video stream. With the evolution of real-time deep learning frameworks such as YOLOv8 (You Only Look Once version 8), object detection has become significantly faster, more accurate, and highly scalable across a range of domains. This advancement has opened up new opportunities in areas like autonomous driving, surveillance, robotics, and smart infrastructure, where understanding and interpreting visual data in real time is crucial. YOLOv8's refined architecture offers lightweight deployment and high detection accuracy, making it well-suited for real-world, latency-sensitive applications.

In this project, object detection is integrated with distance estimation to develop an intelligent assistive system that captures environmental awareness and translates it into real-time audio guidance. The system leverages a live video stream from a camera, processes the input using the YOLOv8 model to detect objects, and computes the distance of each detected object relative to the user. By identifying and announcing the nearest object through audio feedback, the system creates an auditory representation of the visual world, allowing users to sense and respond to their surroundings more intuitively. This seamless conversion of visual to auditory information forms the backbone of the assistive functionality.

A core feature of the solution is its Flask-based web interface, which ensures ease of access and compatibility across different platforms and devices. Users can interact with the system via a browser, removing dependency on specific hardware or operating systems. To further enhance usability, the system incorporates night vision capabilities, enabling it to function effectively even in dim or dark environments. An edge detection mode is also included as a safety enhancement, helping to identify physical boundaries or sudden drops—particularly useful in unfamiliar or risky terrain.

By translating complex visual information into simple, spoken cues, the system offers a meaningful solution for visually impaired individuals, helping them interpret their surroundings without physical assistance. It fosters a greater sense of independence, situational awareness, and personal safety in both indoor and outdoor settings. With its flexible design and real-time responsiveness, this system demonstrates how artificial intelligence and computer vision can be effectively applied to improve quality of life and promote accessibility for all.

Acknowledgement

We would like to express our heartfelt gratitude to all those who have supported and guided us throughout the development of our project, **"**Anecdote: Where Sound Meets Insight."

First and foremost, we extend our sincere thanks to Prof. C.K. Jha, Dean of our institution, for his visionary leadership, continuous encouragement, and unwavering support that inspired us to pursue innovation through this project. His commitment to academic excellence has been a constant source of motivation.

We are equally grateful to Dr. Rajiv Singh, Head of Department, for fostering a learning environment that nurtures creativity and research. His valuable insights and consistent support have played a vital role in the successful completion of our work.

Our heartfelt appreciation goes to our project coordinators, Dr. Neelam Sharma and Dr. Deepak Kumar, whose guidance, timely feedback, and encouragement helped us stay on track and improve our project with every stage of development.

We would especially like to acknowledge the invaluable mentorship of Dr. Khandakar F. Rahman, whose expert advice, constant support, and technical guidance were instrumental in shaping the vision and execution of this project. His mentorship has had a lasting impact on our learning experience.

Lastly, we are thankful to all our faculty members, peers, and family who stood by us throughout this journey. Their support and encouragement were essential in making this project a reality.

1.	2216443	Vibhuti Bhardwaj	CSE
2.	2216446	Vikhyati Singh	CSE
3.	2216319	Mansi Mishra	CSE
4.	2216321	Manya Bhardwaj	CSE

Table of Contents

1. Introduction

Improvements in computer vision, especially object detection and distance measurement, are creating new avenues for assistive technologies to support visually impaired people. These two features are at the core of smart systems that can perceive and describe the visual world, thus providing greater independence, mobility, and safety for the visually impaired.

Object detection refers to detecting and classifying objects within the field of view of a camera—like people, cars, doors, chairs, traffic lights, and other common objects. When combined with wearable devices (such as smart glasses), mobile phones, or even sensor-enabled walking canes, object detection can give immediate feedback regarding the environment of the user. For example, it may alert users to the existence of an oncoming bicycle, a pedestrian crossing, or an open door. This eliminates ambiguity and dependency on others for navigation and interaction with the world.

Adding to this, distance estimation enables the system to quantify how far away from the user each identified object is. This is important for collision avoidance, navigation around obstacles, and decision-making regarding movement. For instance, having information that an object is 0.5 meters away compared to 5 meters can make a big difference in how a user reacts. Whether audio feedback, vibration, or verbal instructions are used, this distance information allows the user to estimate urgency and proximity—information that older technologies such as white canes or guide dogs can't always provide with accuracy.

Together, these technologies provide a more context-aware, responsive, and adaptive way of navigating both indoor and outdoor spaces.

From detecting an unoccupied seat in a waiting room to detecting a staircase or crosswalk signal, these systems translate visual information into usable feedback, allowing users to confidently interpret and engage with the world.

In short, object detection and distance estimation technologies serve as digital vision and spatial understanding, giving the visually impaired person more situational awareness, fewer accidents, and increased independence in daily life.

From now on, AI is transforming mobility for the visually impaired—more safely, more intelligently, and more empowering.

2. Literature Review

2. 1 Image Processing :

- **Residual Blocks and Deep Learning Techniques:** This research focuses on Residual Networks (ResNets), which use shortcut connections to improve deep learning performance and mitigate vanishing gradients in image recognition tasks. Python, with libraries like TensorFlow and PyTorch, supports model implementation, training, and data preprocessing.

Residual Networks (ResNets) are based on the concept of residual learning, which introduces shortcut connections that help mitigate the vanishing gradient problem. The key equation that governs ResNets is:

$$y = F(x) + x$$

where:

a. x is the input to a residual block,

b. F(x) is the learned residual function (i.e., the transformation applied by the neural network layers),

c. y is the output of the residual block. (1)

- **Thresholding Technique for Image Segmentation:** Thresholding is a basic yet effective image segmentation technique that converts grayscale images into binary form by comparing pixel intensities against a set threshold. Pixels above the threshold are marked as foreground; those below as background. This simplifies image analysis by clearly separating objects from the background. Mathematically, this can be expressed as:

$$B(x,y) = \begin{cases} 1, & I(x,y) > T \\ 0, & I(x,y) \leq T \end{cases}$$

where:

- B (x, y) is the binary output image,
- I (x, y) is the intensity of the pixel at coordinates,
- T is the threshold value.

This simple yet effective approach enables the segmentation of objects or regions of interest from the surrounding environment, making it easier to analyze and process these segments individually. In practical applications, such as medical imaging or surveillance, adaptive thresholding methods can be used to dynamically adjust based on local image characteristics:

$$T(x,y) = \mu(x,y) + k \cdot \sigma(x,y)$$

where:

- μ (x, y) is the mean intensity of a local region,
- σ (x, y) is the standard deviation,
- k is a tuning parameter. (2)

- **Granularity in Parallel Computer Vision:**

Granularity in parallel computer vision refers to the degree to which tasks within an image processing pipeline can be decomposed into smaller, independent tasks that can be processed simultaneously across multiple processing units. The speedup achieved by parallelism is often governed by Amdahl's Law:

$$S = \frac{1}{(1-P) + \frac{P}{N}}$$

where:

- S is the speedup,
- P is the proportion of code that can be parallelized,
- N is the number of processing units.

Fine-grained parallelism involves the breakdown of tasks into very small units that can be processed in parallel, allowing for highly efficient computation. By employing parallel architectures, such as multi-core processors or GPUs, this research aims to speed up image processing tasks dramatically. The computational efficiency of parallelism can be expressed as:

$$E = \frac{S}{N}$$

where is the efficiency of the parallel system. Exploring the right granularity of parallelism ensures that computational resources are utilized efficiently without unnecessary overhead. (3)

- **Image denoising:** Image denoising is a crucial step in many computer vision applications, where the goal is to remove noise from an image while preserving important details and features. One effective approach to denoising is using sparse coding, which represents an image as a linear combination of a few basis elements from an over-complete dictionary:

$$I = D\alpha + n$$

where D is an over complete dictionary, α is the sparse coefficient vector, and n represents the noise component. The denoising process involves solving:

$$\min_{\alpha} \|I - D\alpha\|_2^2 + \lambda\|\alpha\|_1$$

where λ is a regularisation parameter controlling sparsity. (4)

- **Thresholding for Image Segmentation:** Thresholding is a basic yet effective image segmentation technique that converts grayscale images into binary form by comparing pixel intensities against a set threshold. Pixels above the threshold are marked as foreground; those below as background. This simplifies image analysis by clearly separating objects from the background. Mathematically, a simple thresholding function can be defined as:

$$I_b(x,y) = \begin{cases} 1, & I(x,y) \geq T \\ 0, & I(x,y) < T \end{cases}$$

where:

- I (x, y) represents the intensity of a pixel at location (x, y),
- T is the threshold value,
- I_b (x, y) is the binary output image where pixels are either foreground (1) or background (0).

For adaptive thresholding, the threshold T (x, y) varies based on the local neighbourhood of the pixel and can be expressed as:

$$T(x,y) = \frac{1}{N} \sum_{(i,j)\in\mathcal{N}} I(i,j)$$

where N represents the local neighbourhood of size N. This ensures dynamic segmentation based on local intensity variations. (5)

2.2 Image Segmentation:

- Deep learning enables precise pixel-level image segmentation using models like U-Net and Mask R-CNN. U-Net's encoder-decoder with skip connections excels in medical imaging, while Mask R-CNN performs instance segmentation by adding a mask branch to Faster R-CNN. These models enhance accuracy in applications like diagnostics, autonomous driving, and object detection. (6)

- **Gradient-based segmentation in image processing and computer vision:** Gradient-based segmentation is a key technique in computer vision that detects object boundaries by analyzing intensity changes (gradients) in an image. It identifies edges where pixel values shift significantly, helping to delineate objects and regions. This method is especially useful in food quality assessment, where it can detect defects like bruises or discoloration by highlighting texture or shape changes. By enabling precise, automated inspection of food surfaces, gradient-based segmentation enhances quality control, reduces human error, and increases efficiency in production lines. Mathematically, the image gradient at a pixel (x, y) is given by the vector:

$$\nabla I(x,y) = \left[\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right]$$

The gradient magnitude, which indicates the strength of an edge at a point, is computed as:

$$|\nabla I(x,y)| = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}$$

The gradient direction (angle of the edge) is given by:

$$\theta(x, y) = \tan^{-1}\left(\frac{\partial I/\partial y}{\partial I/\partial x}\right)$$

These calculations are commonly implemented using Sobel, Prewitt, or Scharr filters to approximate the partial derivatives. (7)

- **Gaussian smoothing:** Gradient-based segmentation detects object boundaries by analyzing changes in pixel intensity, making it ideal for edge detection and structure recognition. In food quality assessment, this technique helps identify defects like bruises or discoloration by highlighting texture and shape variations. It enables precise, automated inspection, improving consistency and efficiency in quality control. With advancements in machine learning, gradient-based methods are becoming more accurate and adaptable for real-world applications. The Gaussian smoothing filter is mathematically defined by the Gaussian function:

$$G(x, y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

Where:

- x and y are the pixel coordinates,
- σ is the standard deviation of the Gaussian distribution, determining the amount of smoothing.

The smoothed image IsIs is obtained by convolving the original image II with the Gaussian kernel G:

$$I_s(x, y) = (I * G)(x, y) = \sum_{i=-k}^{k}\sum_{j=-k}^{k} I(x - i, y - j) \cdot G(i, j)$$

This operation effectively reduces high-frequency noise while maintaining the important structural content of the image. (8)

- **Generative Adversarial Networks (GANs):** GANs, consisting of a generator and discriminator, produce realistic synthetic images through an adversarial training process. They enhance image classification by augmenting datasets—especially useful when labelled data is scarce, such as in medical imaging. In object detection, GANs generate varied training samples (e.g., different lighting, angles), improving model precision in real-world scenarios like autonomous driving. For segmentation tasks, GANs create high-resolution, context-aware images, helping models detect fine details where annotated data is limited. Overall, GANs boost generalization by enriching training data with diverse, customizable samples, improving performance across classification, detection, and segmentation.

Mathematical Representation of GANs:

A GAN is composed of two models: a generator and a discriminator, where:

- $z \sim p_z(z)$: input noise vector to the generator
- $x \sim p_{data}(x)$: real data sample
- G(z): generated (fake) data
- D(x): probability that xx is from the real data rather than G(z)

The GAN objective function is defined as:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

The generator tries to minimise this function, while the discriminator tries to maximise it. (9)

- **Level Set Methods and Distance Regularized Level Set Evolution (DRLSE) for Image Segmentation**: Level set methods segment images by

evolving contours to outline object boundaries. Distance Regularized Level Set Evolution (DRLSE) enhances this process by maintaining smooth, stable contours through a distance regularization term, preventing distortion or shrinkage. DRLSE excels in segmenting complex or ambiguous boundaries, making it ideal for applications like medical imaging and satellite analysis where accuracy is critical.

Mathematical Expression of Level Set and DRLSE:

Let ϕ (x, y, t) be a level set function representing the evolving contour as its zero-level set.

The general level set evolution equation is:

$$\frac{\partial \phi}{\partial t} = F|\nabla \phi|$$

Where:

- ϕ is the level set function
- F is the speed function driving the contour evolution
- ∇ϕ is the gradient of the level set function

In DRLSE, the energy functional is given by:

$$E(\phi) = \mu R(\phi) + \lambda L(\phi) + \alpha A(\phi)$$

Where:

- R(ϕ): distance regularization term (maintains regularity of the level set function)
- L(ϕ): length term (encourages the contour to align with object boundaries)
- A(ϕ): area term (controls the contour's expansion or shrinkage)

- μ, λ, α are weighting parameters that control the influence of each term

The distance regularization term R(ϕ) ensures that ϕ maintains a desirable signed distance property during the evolution, improving both numerical stability and segmentation accuracy. (10)

- **U-Net for Semantic Image Segmentation:** U-Net is a widely used architecture for semantic segmentation, featuring an encoder-decoder structure with skip connections. The encoder captures global context by downsampling the image, while the decoder reconstructs detailed segmentation maps using high-resolution features from the encoder. This design preserves both global and fine-grained information, enabling accurate segmentation in complex scenarios like medical imaging, satellite analysis, and autonomous driving. Its flexibility with varying image sizes makes U-Net a robust and versatile choice for precise segmentation tasks.

  Mathematical Representation of U-Net Components:

  Let $I \in \mathbb{R}^{H \times W \times C}$ be the input image.

  a. **Encoder Operation:** Each layer of the encoder applies convolution followed by non-linearity and downsampling:

  $$f^{(l)} = \text{ReLU}(\text{Conv}(f^{(l-1)}))$$

  Downsampling (usually max pooling) reduces spatial resolution:

  $$f^{(l)}_{\text{down}} = \text{MaxPool}(f^{(l)})$$

  b. **Skip Connections:** Feature maps from the encoder are directly passed to the decoder:

  $$s^{(l)} = f^{(l)}_{\text{encoder}}$$

c. **Decoder Operation:** Decoder layers upsample the feature maps and concatenate with corresponding encoder maps:

$$f_{\text{up}}^{(l)} = \text{UpSample}(f^{(l+1)}) \oplus s^{(l)}$$

Then apply convolution to refine features:

$$f^{(l)} = \text{ReLU}(\text{Conv}(f_{\text{up}}^{(l)}))$$

d. **Final Output:** The final layer applies a softmax or sigmoid depending on the segmentation task:

$$\hat{Y} = \sigma(\text{Conv}_{1\times1}(f^{(0)}))$$

where $\hat{Y}$ is the predicted segmentation map. (11)

- **Granulometry and Morphological Operations for Texture and Particle Size Analysis:** Granulometry uses morphological operations to analyze particle size and texture in images by applying structuring elements that highlight specific features. It helps distinguish objects from noise, enhancing segmentation accuracy in complex images. Commonly used in materials science, biology, and environmental monitoring, this technique improves texture understanding and supports robust image analysis by identifying the shape, size, and distribution of particles.

  Mathematical Expressions for Granulometry:

  Let f(x) be a grayscale image and B_r a structuring element of radius r.
  The granulometric profile is defined based on the
  morphological opening operation:

$$\gamma_{B_r}(f) = (f \circ B_r)$$

  where ∘∘ denotes the morphological opening:

$$f \circ B_r = \delta_{B_r}(\epsilon_{B_r}(f))$$

Here:

- ϵ_{Br} (f) Erosion of f by B_r
- δ_{Br}: Dilation of the eroded image

The granulometric size distribution function is then calculated as:

$$G(r) = \int_{\Omega} f(x)\,dx - \int_{\Omega} \gamma_{B_r}(f(x))\,dx \quad (12)$$

- **U-Net, Mask R-CNN, and GANs for Enhanced Image Segmentation and Object Detection:** U-Net uses an encoder-decoder structure with skip connections to capture fine details, making it ideal for precise image segmentation. Mask R-CNN enhances object detection by adding a mask prediction branch for instance-level segmentation. GANs support the process by generating synthetic images to enrich training data, boosting model robustness and accuracy. Together, they create a strong pipeline for high-quality segmentation in complex applications like medical imaging and autonomous driving. (13)

- **U-Net, Mask R-CNN, and GANs for Medical Image Segmentation:** U-Net excels in segmenting organs and tumors by preserving both context and fine details. Mask R-CNN enables instance-level segmentation, handling overlapping structures in medical images. GANs boost performance by generating synthetic training data, making the system more robust in rare or low-data scenarios. (14)

- **YOLO for Real-Time Object Detection in Earth Observation:** YOLO enables real-time object detection by processing satellite or aerial images in a single pass, making it fast and efficient. Ideal for land use monitoring, environmental tracking, and wildlife detection, YOLO supports large-scale, time-sensitive analysis in earth observation.
  Mathematical Expressions for YOLO:

  YOLO divides the input image into an $S \times S$ grid. Each grid cell predicts:

  - B bounding boxes
  - Confidence score:

$$\text{Confidence} = P(\text{object}) \times \text{IOU}_{\text{pred}}^{\text{truth}}$$

  Each bounding box prediction includes:

$$(x, y, w, h, \text{confidence})$$

  The final output vector per cell:

$$\text{Prediction} = (x, y, w, h, \text{confidence}, p_1, p_2, \ldots, p_c)$$

  Where $p_i = P(\text{class}_i \mid \text{object})$ for c classes.

  YOLO's loss function combines localization loss, confidence loss, and classification loss:

$$\text{Loss} = \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] + \ldots$$

  This formulation enables YOLO to learn object detection in a single unified architecture. (15)

2.3 Object And Edge Detection:

- **TensorFlow and PyTorch for CNNs in Thermal Infrared Image Processing:** TensorFlow and PyTorch enable powerful CNN-based processing of thermal infrared images, tackling challenges like low contrast and noise. They support advanced model design, large dataset handling, and accurate object and edge detection, making them ideal for surveillance, autonomous vehicles, and military reconnaissance.

  Mathematical Expressions for CNN-based Processing:

  A typical convolution operation in CNNs is expressed as:

$$y(i,j) = \sum_m \sum_n x(i+m, j+n) \cdot k(m,n)$$

  Where:

  - x (i, j) is the input image,
  - k (m, n) is the kernel (filter),
  - y (i, j) is the output feature map.

  The activation function used after convolution (commonly ReLU):

$$f(x) = \max(0, x)$$

  The loss function (e.g., cross-entropy for classification):

$$L = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$

  Where:

  - y_i is the true label,
  - $\hat{y}_i$ is the predicted probability.

  Backpropagation updates weights using gradient descent:

$$w := w - \eta \frac{\partial L}{\partial w}$$

Where:

- w is the weight,
- η is the learning rate.

These formulations are efficiently implemented and managed in both TensorFlow and PyTorch environments to optimize thermal image analysis.

(16)

- **TensorFlow/Keras for Small Infrared Target Detection:** TensorFlow/Keras enables the development of deep learning models for detecting small, low-contrast infrared targets. By using custom layers and thermal-specific features, it enhances visibility and accuracy in challenging environments. Ideal for defense, search and rescue, and surveillance applications.

(17)

- **YOLOv5 Enhanced to YOLO-FIRI for Infrared Object Detection:** YOLO-FIRI is an improved version of YOLOv5, optimized for infrared images. It features enhanced feature extraction and fusion to handle low-contrast, noisy thermal data, enabling accurate real-time detection. Ideal for search and rescue, wildlife monitoring, and military surveillance in low-visibility conditions.

  Mathematical Expressions for YOLO-FIRI Object Detection:

  YOLO predicts bounding boxes as:

$$t_x = \sigma(\hat{t}_x) + c_x, \quad t_y = \sigma(\hat{t}_y) + c_y$$

$$t_w = p_w e^{\hat{t}_w}, \quad t_h = p_h e^{\hat{t}_h}$$

  Where:

- $\hat{t}_x$, $\hat{t}_y$, $\hat{t}_w$, $\hat{t}_h$ are predicted offsets,
- c_x are the coordinates of the grid cell,
- p_w p_h are anchor box dimensions.

The objectness score is calculated as:

$$P_{obj} = \sigma(\hat{P}_{obj})$$

Total loss function used in YOLO-FIRI, combining classification, objectness, and bounding box regression:

$$\mathcal{L} = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (w_i - \hat{w}_i)^2 + (h_i - \hat{h}_i)^2 \right] + \lambda_{obj} \sum (\text{Objectness Loss}) + \lambda_{cls} \sum (\text{Classification Loss})$$

These formulations ensure YOLO-FIRI can learn and adapt to the nuances of thermal imagery, making it highly suitable for real-time object detection in infrared conditions. (18)

- **R-CNN for Object Detection in UAV-Captured Infrared Images:** R-CNN excels at detecting objects in infrared images captured by UAVs, handling varied scales and complex backgrounds. It generates region proposals, then classifies and refines them using CNNs, enabling accurate detection in aerial thermal imagery. Ideal for surveillance, environmental monitoring, and disaster response.

  Mathematical Expressions for R-CNN Object Detection:

  a. Region Proposal:

     Let II be the input image. The selective search algorithm generates NN region proposals $R=\{r_1, r_2,\ldots,r_N\}$R={r1,r2,…,rN}, where each ri⊂I.

  b. Feature Extraction using CNN:

     For each proposed region r_i, features are extracted using a convolutional neural network:

$$f_i = \text{CNN}(r_i)$$

c. Classification and Bounding Box Regression:
Classification score:
Bounding box refinement:

$\Delta bi = Wrfi + br$

where $\Delta bi = (\Delta x, \Delta y, \Delta w, \Delta h)$ represents the adjustments to the proposed bounding box.

d. Loss Function:
The R-CNN loss combines classification and bounding box regression:

$$\mathcal{L} = \mathcal{L}_{cls}(p_i, p_i^*) + \lambda \cdot 1_{\{p_i^* > 0\}} \cdot \mathcal{L}_{reg}(\Delta b_i, \Delta b_i^*)$$

where:

- p_i^* is the ground truth label,
- Δb_i^* is the ground truth bounding box adjustment,
- λ balances the two losses.

These mathematical components help R-CNN models to accurately detect and localize objects even in noisy, low-contrast infrared images from aerial platforms.
(19)

- **Deepcham's Mobile Object Recognition with ONNX and TensorFlow:** Deepcham leverages ONNX for cross-platform model interoperability, enabling seamless deployment of models trained in frameworks like TensorFlow or PyTorch. This flexibility supports efficient real-time object recognition on mobile and edge devices. TensorFlow handles training and inference, while ONNX ensures smooth adaptation across platforms for collaborative edge computing.

Mathematical Expressions for Model Interoperability and Mobile Object Recognition:

Let f (x; θ) be a deep learning model trained in TensorFlow with parameters θ. Using ONNX, this model is converted to an equivalent representation $\tilde{f}(x;\theta')$ such that:

$$f(x;\theta) \approx \tilde{f}(x;\theta')$$

For mobile object recognition, the final classification layer typically computes the predicted class probabilities as:

$$\hat{y} = \text{softmax}(Wx + b)$$

where:

a. x represents the extracted features from the input image,
b. W and b are the weight matrix and bias vector,
c. $\hat{y}$ denotes the vector of class probabilities.

These formulations ensure that Deepcham's recognition system maintains consistent performance across various platforms through effective model interoperability and robust deep learning processing. (20)

- **Real-Time Object Detection on Mobile Devices Using TensorFlow Lite:**

TensorFlow Lite enables fast, efficient real-time object detection directly on mobile devices by optimizing deep learning models for low-power environments. It reduces model size and computation without sacrificing performance, making it ideal for AR, robotics, and surveillance, where low latency and quick responses are crucial. Mathematical Expressions for Mobile Object Detection with TensorFlow Lite:

A typical deep learning model for object detection predicts bounding boxes and class probabilities from an input image I. For an optimized model running on TensorFlow Lite, the model function can be represented as:

$$\hat{Y} = f(I; \theta)$$

Where:

- I is the input image,
- θ denotes the optimized model parameters,
- $\hat{Y}$ is the model output that includes bounding box coordinates and class probabilities.

Each predicted bounding box is represented as:

$$\text{BBox} = (x, y, w, h, c)$$

Where:

- (x, y) denote the center coordinates,
- (w, h) represent the width and height,
- c is the confidence score.

The final classification for each bounding box is typically obtained using a softmax function:

$$\hat{p}_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

Where:

a. z_i are the logits for each class i,
b. $\hat{p}_i$ is the predicted probability for class i.

These formulations, when deployed on TensorFlow Lite, enable efficient and accurate real-time object detection on mobile devices with minimal latency. (21)

- **Patch-Based Approach for Small Target Detection in Infrared Images Using Scikit-Image:** This approach divides infrared images into small patches for localized detection of tiny targets like vehicles or people. Using Scikit-Image, it efficiently processes each patch with tools for feature extraction and noise handling. Ideal for real-time, mobile applications like surveillance and monitoring, especially in noisy thermal environments.

  Mathematical Expressions for Patch-Based Small Target Detection:

  Let the input infrared image be represented as:

$$I \in \mathbb{R}^{H \times W}$$

  We divide III into a set of overlapping or non-overlapping patches $P_k \in \mathbb{R}^{h \times w}$, where:

$$I = \bigcup_{k=1}^{N} P_k$$

  Each patch P_k is analysed using a detection function f, parameterized by a model θ:

$$\hat{y}_k = f(P_k; \theta)$$

  Where:

- $\hat{y}_k$ is the predicted output for patch P_k indicating the presence or absence of a small target.

  To reconstruct the detection map for the entire image, the outputs from all patches are aggregated:

$$\hat{Y} = \sum_{k=1}^{N} \hat{y}_k \cdot \mathbb{I}_{P_k}$$

Here, $\mathbb{I}_{P_k}$ is an indicator function that maps each patch prediction back to its location in the original image. (22)

- **UIU-Net for Small Object Detection in Infrared Images:** UIU-Net uses a nested U-Net architecture to enhance small object detection in infrared images. By extracting features at multiple scales, it captures both fine details and broader context, improving accuracy in noisy or low-contrast environments. It's especially effective for real-time infrared tasks like search and rescue, surveillance, and wildlife monitoring.

  Mathematical Expressions for UIU-Net Architecture:

  Let the input image be:

$$I \in \mathbb{R}^{H \times W}$$

  The first U-Net encoder extracts hierarchical features at multiple scales:

$$F^1 = \text{Encoder}_{U1}(I)$$

  These features F1F^1F1 are passed to an internal U-Net:

$$F^2 = \text{U-Net}_{inner}(F^1)$$

  The decoder of the outer U-Net then reconstructs the final segmentation map:

$$\hat{Y} = \text{Decoder}_{U1}(F^2)$$

  Where:

- $\hat{Y} \in \mathbb{R}^{H \times W}$ is the final prediction map highlighting small object locations.

  The nested structure allows for deeper feature fusion:

$$\hat{Y} = \text{Decoder}_{U1}(\text{U-Net}_{inner}(\text{Encoder}_{U1}(I)))$$

This multilevel abstraction enhances the detection of small infrared targets by fusing fine-grained spatial features with high-level semantic information.

(23)

- **YOLO-ACN for Improved Detection of Small and Occluded Objects:** YOLO-ACN is an improved YOLO model that enhances detection of small and partially occluded objects. It integrates attention mechanisms to focus on key image areas and context networks to understand object-environment relationships. This makes YOLO-ACN ideal for crowded scenes, surveillance, and tasks requiring detection of fine details.

  The YOLO-ACN enhancements can be expressed concisely as follows:

  a. Feature Extraction:

$$F = \phi_{\text{backbone}}(I)$$

  b. Attention Mechanisms:

    - Spatial Attention:

$$M_s = \sigma(f_s(F)), \quad F' = M_s \odot F$$

    - Channel Attention:

$$M_c = \sigma(f_c(F)), \quad F'' = M_c \odot F'$$

  c. Context Network for Object Relationships:

$$F_{\text{context}} = \phi_{\text{context}}(F'')$$

  d. Final Object Detection:

$$\hat{Y} = \phi_{\text{detector}}(F_{\text{context}})$$

  Where $\hat{Y} = (x, y, w, h, p_{\text{class}})$. (24)

2.4 Distance Estimation:

- **OpenCV for 3D Object Reconstruction from Mobile Images**

  OpenCV enables 3D object reconstruction from mobile images by using key computer vision techniques like feature detection, feature matching, and depth estimation. These tools align multiple 2D images and extract depth information to build accurate 3D models. This transforms mobile devices into effective 3D scanners, useful in augmented reality, digital content creation, and virtual environments.

  The key mathematical formulations are:

a. Feature Detection and Matching

Feature points are detected in two images I_1 and I_2, and matched using descriptors (e.g., SIFT, ORB):

$$\text{Matches} = \text{match}(\text{features}(I_1), \text{features}(I_2))$$

b. Fundamental Matrix Estimation

Given corresponding points (x_1, y_1) in $I_1$and (x_2, y_2) in I_2, the Fundamental Matrix F satisfies:

$$[x_2, y_2, 1]F[x_1, y_1, 1]^T = 0$$

F is estimated using the Eight-Point Algorithm**.**

c. Essential Matrix Computation

If camera intrinsic matrix K is known:

$$E = K^T F K$$

where E encodes the relative pose between the two camera views. (25)

- **Zoe Depth Estimation for Mobile Monocular 3D Object Detection:**
  Zoe Depth Estimation improves monocular 3D object detection on mobile devices by using ground plane priors to enhance depth accuracy from a single camera. Unlike stereo systems, it estimates object distance and size more reliably by referencing ground depth, leading to more consistent and accurate results. This technique is especially valuable for real-time applications like autonomous driving, robotics, and AR, where precise 3D understanding is critical.

  a. Simplified Formulas for Zoe Depth Estimation in Mobile Monocular 3D Object Detection:

  Zoe Depth Estimation improves monocular depth perception using a ground plane prior for better accuracy. The key equations are:

  b. Depth Prediction from a Single Image

  A deep network ϕ_{depth} estimates the depth D from an image I:

$$D = \phi_{\text{depth}}(I)$$

  c. Ground Plane Prior Integration

  The estimated depth is refined using a ground plane constraint G:

$$D' = D + G$$

  where G adjusts depth based on known ground features.

  d. 3D Object Position Estimation

  Given camera intrinsic matrix KKK, the 3D location P (X, Y, Z) of an object detected at pixel (x, y) is:

$$D' = D + G \tag{26}$$

- **OpenCV in Preprocessing and Post-Processing for Monocular Depth Estimation:** OpenCV enhances monocular depth estimation by supporting effective preprocessing and post-processing. Since depth is inferred from a single image (without stereo cues), preprocessing is vital for improving model accuracy.

  Preprocessing: OpenCV handles image resizing, normalization, and filtering to prepare inputs for the model. Feature extraction (e.g., edges, corners) and data augmentation (e.g., rotation, flipping) boost model generalization and robustness.
  Post-Processing: After prediction, OpenCV refines depth maps by reducing noise, smoothing errors, and enhancing consistency using filtering and image transformation techniques.

  By integrating OpenCV, depth estimation becomes more accurate and efficient, supporting applications like 3D reconstruction, robotic navigation, and autonomous driving.

a. Preprocessing: Image Transformation

   To standardize input images I, OpenCV applies transformations:

$$I' = \phi_{\text{pre}}(I)$$

   where ϕ_{pre} includes resizing, normalization, and filtering.

b. Post-Processing: Depth Refinement

   After predicting depth D, OpenCV refines it using smoothing filters:

$$D' = \phi_{\text{post}}(D)$$

   where ϕ_{post} applies bilateral filtering, median filtering, or guided filtering to reduce noise and enhance depth accuracy. (27)

3. Methodology

3.1 Image Processing

Image Processing is one of computer vision and digital signal processing concerned with the analysis, enhancement, and manipulation of images. It's applied in medical imaging, satellite imaging, face recognition, and self-driving cars.

- **Thresholding:**

Function: Splits pixels within an image into separate categories (for example, background and foreground) according to intensity values.

Implementation:

Often employed for binary segmentation, where pixels over a given value are labeled with one value (e.g., 255), and pixels with values below this are labeled with another (e.g., 0).

- **Granularity:**

Purpose: Refers to the resolution of detail or task partition in parallel computing. In image processing, granularity will define how small tasks are partitioned for concurrent execution (fine or coarse).

Implementation:

Lower granularity enables parallel processing of smaller image blocks or features, enhancing computational efficiency.

- **Image Denoising:**

Purpose: Eliminates noise from an image without degrading significant structures and details.

Implementation:

Techniques such as wavelet transform, Gaussian filtering, or sparse representations are employed to minimize noise.

- **Texture Analysis:**

Purpose: Identifies patterns, repetitions, and surface characteristics of objects in an image, commonly applied for segmentation or classification.

Implementation:

Methods such as Gray-Level Co-occurrence Matrix (GLCM) or Local Binary Patterns (LBP) measure texture features like contrast, homogeneity, and entropy.

Every method adds to the image processing pipeline—from quality improvement and segmentation to efficient computation and feature extraction.

3.2 Segmentation

Segmentation separates an image into significant regions to distinguish objects from the background, making analysis easier and supporting object recognition.

- **UNet:**

Purpose: A convolutional neural network structure for biomedical image segmentation.

Implementation:

UNet consists of a symmetric encoder-decoder architecture in which the encoder learns features, and the decoder outputs the segmented image with accurate localization through skip connections.

- **Mask R-CNN:**

Purpose: A variant of Faster R-CNN for instance segmentation, which detects objects and, in addition, produces a pixel-level mask for every object.

Implementation:

Makes use of a Region Proposal Network (RPN) to propose objects and segmentation masks for pixel-wise classification.

- **GANs (Generative Adversarial Networks):**

Purpose: Utilized for unsupervised segmentation by producing realistic segmented images.

Implementation:

A GAN has a generator (produces new images) and a discriminator (distinguishes between real and generated images). Models based on GAN can produce segmentation masks in a self-supervised manner.

- **Gradient-based Segmentation:**

Purpose: Segment images according to changes in intensity, detecting edges by searching for prominent changes in gradients.

Implementation:

Uses operators like Sobel or Canny edge detectors to detect gradients and segment the image based on edges.

- **Gaussian Smoothing:**

Purpose: A pre-processing step that smoothens images, reducing noise and improving the segmentation of smooth regions.

Implementation:

Applies a Gaussian filter to blur the image, making it easier to apply thresholding or edge detection methods.

- **Level Set Methods:**

Purpose: Deform a contour over time to identify object boundaries from minimizing energy functions.

Implementation:

A deformable model in which a curve (level set) is updated iteratively to match the object boundary, permitting topology changes such as merging or splitting during segmentation.

- **Granulometry:**

Purpose: A mathematical method for estimating object size distributions in an image.

Implementation:

Applies morphological operations (e.g., dilation and erosion) to filter objects by size, commonly applied to eliminate noise or segment objects according to size.

- **Morphological Filtering:**

Purpose: Refines segmentation by eliminating noise or smoothing boundaries of objects with operations such as erosion, dilation, opening, and closing.

Implementation:

Morphological operations are used to refine segmentation masks, eliminating small objects or closing gaps.

- **YOLO (You Only Look Once):**

Purpose: A real-time object detection model that also offers segmentation through other modifications (e.g., YOLO-segmentation).

Implementation:

YOLO is able to predict bounding boxes and classify objects in a single pass, which is very efficient for real-time tasks.

Each of these methods has its own advantages, UNet is best suited for pixel-level accuracy, Mask R-CNN for instance segmentation, and YOLO for real-time detection. They are selected depending on the task and application requirements.

3.3 Object Recognition/Detection

Object detection and object recognition are fundamental computer vision tasks that enable machines to recognize and pinpoint objects within images or video. They are critical in autonomous vehicles, face recognition, medical imaging, automation in industry, and AR. New machine learning methods make it possible to achieve high precision and efficiency in these tasks. Below is a brief summary of the best methods

- **YOLO (You Only Look Once) v8**

Purpose: YOLO is among the fastest real-time object detection systems, able to detect multiple objects in a single pass of an image.

Implementation:

YOLOv8 can be utilized for high-speed detection, using pre-trained weights. Fine-tuning the model on a specific dataset enables it to detect particular objects specific to the application.

- **R-CNN (Region-based Convolutional Neural Networks)**

Purpose: R-CNN enhances object detection accuracy by suggesting Regions of Interest (RoIs) prior to using CNNs for classification.

Implementation:

Faster R-CNN: A major advance over the standard R-CNN, Faster R-CNN combines a Region Proposal Network (RPN) to make it more efficient.

Mask R-CNN: Expands Faster R-CNN with the addition of segmentation masks to provide accurate pixel-level object detection.

- **U-Net for Object Detection**

Purpose: Although U-Net is mostly applied for image segmentation, it can also detect objects at a pixel level.

Implementation:

U-Net's encoder-decoder architecture is perfectly suited to detect objects with high accuracy and can be applied in medical imaging and satellite imagery analysis.

Granulometry and Morphological Filtering (Noise Reduction)

Purpose: Granulometry is used to estimate the size distribution of objects in an image, and morphological filtering strengthens object boundaries and eliminates noise.

Implementation:

Granulometry: Applied to study the structure of granular materials, e.g., cells in a microscope image.

Morphological Operations: Operations such as erosion and dilation clean up object boundaries and eliminate small, insignificant objects.

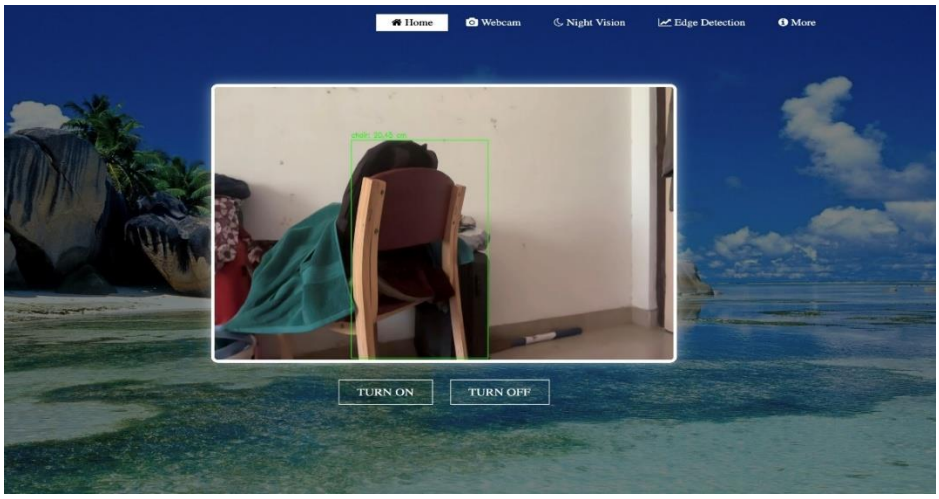
3.4 Distance Estimation

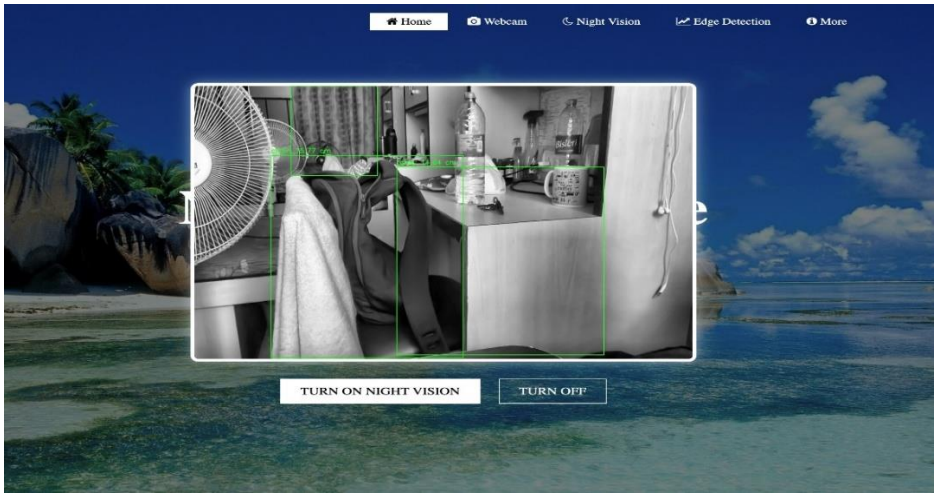
Distance estimation calculates how far away objects are from a sensor or camera, allowing systems to comprehend depth and spatial relationships—important in robotics, self-driving cars, AR, and computer vision.

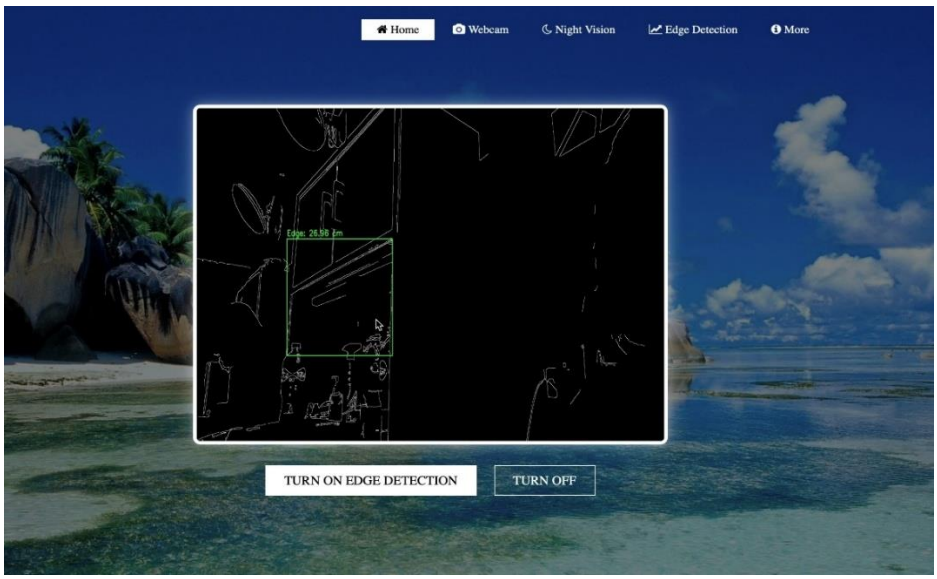
- **OpenCV**: OpenCV is a popular computer vision library for image processing and distance measurement via stereo vision, camera calibration, and depth mapping.
    a. Stereo Vision: Utilizes two cameras and disparity maps to estimate depth through triangulation.
    b. Single Camera Calibration: Uses models (e.g., pinhole camera) to connect pixel information with real-distance.
    c. Edge & Contour Detection: Detects object edges and contours to aid in distance measurement.

OpenCV accommodates conventional techniques, whereas deep learning software such as Zoe facilitates single-image depth estimation—both are useful for real-world purposes.

3.5 User Interface and Code Segments:







```python
from flask import Flask, Response, render_template, jsonify
from flask_cors import CORS
import cv2
import time
import os
from gtts import gTTS
from ultralytics import YOLO

app = Flask(__name__)
CORS(app)

model = YOLO('/Users/vibhutibhardwaj/runs/detect/train4/weights/best.pt')

KNOWN_WIDTH = 7.0
FOCAL_LENGTH = 1671.43

LANGUAGE_MAP = {"English": "en", "Hindi": "hi", "French": "fr", "Spanish": "es", "German": "de"}
SELECTED_LANGUAGE = "English"

camera = None
streaming = False
```

```python
def estimate_distance(known_width, focal_length,
def speak(text, lang="en"): …
def generate_frames(): …
def generate_frames_night(): …
def generate_frames_edge(): …
@app.route('/start_feed')
def start_feed(): …
@app.route('/stop_feed')
def stop_feed(): …
@app.route('/video_feed')
def video_feed(): …
@app.route('/video_feed_night')
def video_feed_night(): …
@app.route('/video_feed_edge')
def video_feed_edge(): …
@app.route('/start_edge_detection')
def start_edge_detection(): …
```

```
0: 384x640 1 chair, 1 table, 1 tree, 87.3ms
Speed: 1.2ms preprocess, 87.3ms inference, 0.7ms postprocess per image at sha
pe (1, 3, 384, 640)

0: 384x640 1 chair, 1 table, 1 tree, 87.2ms
Speed: 1.2ms preprocess, 87.2ms inference, 0.6ms postprocess per image at sha
pe (1, 3, 384, 640)

0: 384x640 1 chair, 1 table, 1 tree, 87.3ms
Speed: 1.2ms preprocess, 87.3ms inference, 0.5ms postprocess per image at sha
pe (1, 3, 384, 640)

0: 384x640 1 chair, 1 table, 1 tree, 89.4ms
Speed: 1.2ms preprocess, 89.4ms inference, 0.5ms postprocess per image at sha
pe (1, 3, 384, 640)

0: 384x640 1 chair, 1 table, 1 tree, 90.5ms
Speed: 1.2ms preprocess, 90.5ms inference, 0.5ms postprocess per image at sha
pe (1, 3, 384, 640)

0: 384x640 1 chair, 1 table, 1 tree, 101.1ms
Speed: 1.2ms preprocess, 101.1ms inference, 0.6ms postprocess per image at sh
ape (1, 3, 384, 640)

0: 384x640 1 chair, 1 table, 1 tree, 162.0ms
Speed: 1.4ms preprocess, 162.0ms inference, 1.1ms postprocess per image at sh
ape (1, 3, 384, 640)

0: 384x640 1 chair, 1 table, 1 tree, 126.0ms
Speed: 2.1ms preprocess, 126.0ms inference, 0.6ms postprocess per image at sh
ape (1, 3, 384, 640)
0: 384x640 1 chair, 1 table, 1 tree, 104.1ms
Speed: 2.3ms preprocess, 104.1ms inference, 0.8ms postprocess per image at sh
ape (1, 3, 384, 640)

0: 384x640 1 chair, 1 table, 1 tree, 90.1ms
Speed: 1.3ms preprocess, 90.1ms inference, 0.5ms postprocess per image at sha
pe (1, 3, 384, 640)

0: 384x640 1 chair, 1 table, 89.0ms
Speed: 1.4ms preprocess, 89.0ms inference, 0.5ms postprocess per image at sha
pe (1, 3, 384, 640)

0: 384x640 1 chair, 1 table, 1 tree, 96.1ms
Speed: 1.3ms preprocess, 96.1ms inference, 0.7ms postprocess per image at sha
pe (1, 3, 384, 640)

0: 384x640 1 chair, 1 table, 1 tree, 88.5ms
Speed: 1.2ms preprocess, 88.5ms inference, 0.6ms postprocess per image at sha
pe (1, 3, 384, 640)

0: 384x640 1 chair, 1 table, 1 tree, 89.6ms
Speed: 1.3ms preprocess, 89.6ms inference, 0.5ms postprocess per image at sha
pe (1, 3, 384, 640)
Speaking: table detected at 14 centimeters in en
127.0.0.1 - - [06/Apr/2025 12:53:51] "GET /stop_feed HTTP/1.1" 200 -
```

4. Results and Discussion

4.1 Noise Reduction: Noise reduction is a crucial aspect of image processing that aims to remove or minimize unwanted variations in an image while preserving important details. Various sources, such as sensor limitations, transmission errors, or environmental factors, introduce noise into images. Effective noise reduction techniques help enhance image quality for better interpretation and analysis.

This document presents a comparative analysis of three common filtering techniques: Bilateral Filtering, Gaussian Filtering, and Median Filtering, focusing on their effectiveness in noise reduction based on PSNR (Peak Signal-to-Noise Ratio) and SSIM (Structural Similarity Index).

A. Bilateral Filtering:

- Maintains sharp edges while reducing noise.
- Uses both spatial proximity and intensity similarity for smoothing.
- Produces the highest PSNR (≈ 50-55 dB) and most stable SSIM (≈ 1.0 after index 6).
- Best overall performance for noise reduction.

B. Gaussian Filtering:

- Uses a weighted average based on a Gaussian function.
- Smooths images but can blur edges.
- Moderate noise reduction performance with fluctuating PSNR (≈ 22-26 dB) and SSIM (≈ 0.6-0.8 after index 6).

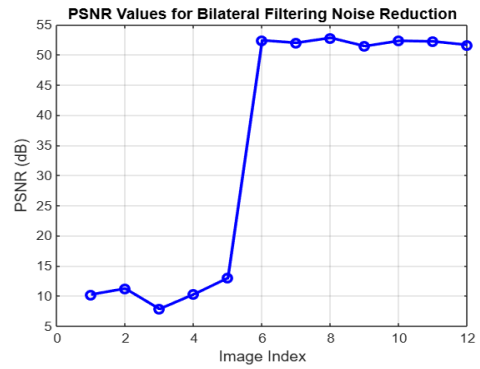
C. Median Filtering:

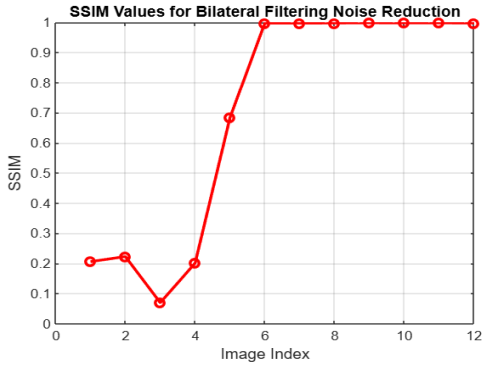
- Replaces each pixel's value with the median of its neighborhood.
- Effective in reducing salt-and-pepper noise while preserving edges.

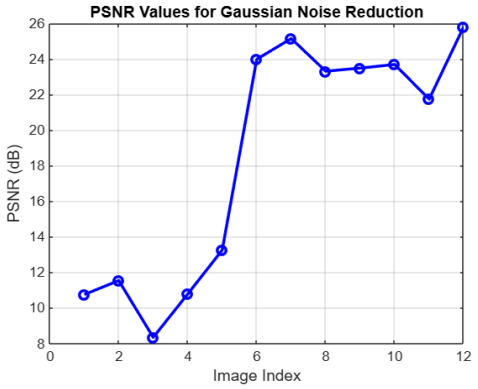
- Produces high PSNR (≈ 35-39 dB) and stable SSIM (≈ 0.9-1.0 after index 6).
- Performs better than Gaussian filtering but is not as effective as bilateral filtering.

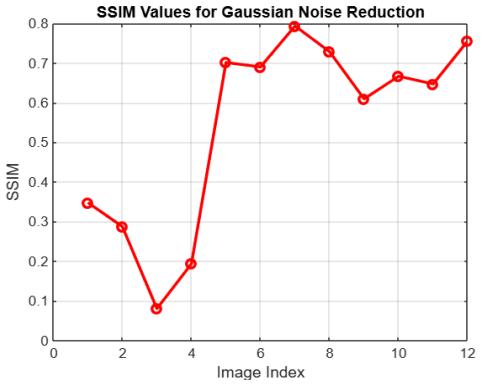
Comparative Analysis:

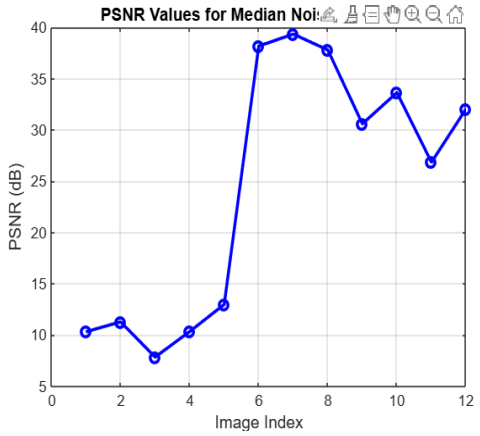
Metric	Bilateral Filtering	Gaussian Filtering	Median Filtering
PSNR (Peak Signal-to-Noise Ratio) Improvement	Highest (≈ 50-55 dB)	Moderate (≈ 22-26 dB)	High (≈ 35-39 dB)
PSNR Stability	Very stable after initial rise	Fluctuates slightly after peak	Moderate fluctuations after peak
SSIM (Structural Similarity Index) Improvement	Best (≈1.0)	Moderate (≈ 0.6-0.8 after index 6)	High (≈ 0.9-1.0 after index 6)
SSIM Stability	Very stable, close to 1.0	Noticeable fluctuations	Some variations but close to 1.0
Initial PSNR & SSIM Before Filtering	Low for all (PSNR ≈ 10-15 dB, SSIM ≈ 0.1-0.3)	Low for all (PSNR ≈ 10-15 dB, SSIM ≈ 0.1-0.3)	Low for all (PSNR ≈ 10-15 dB, SSIM ≈ 0.1-0.3)
Performance on Noise Reduction	Best overall	Moderate performance	Better than Gaussian but not as good as Bilateral

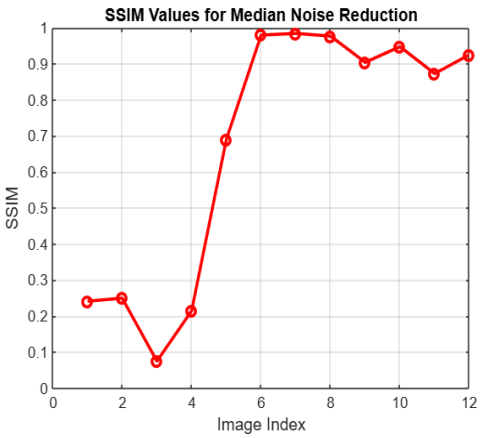












The comparison of the three filtering techniques is based on PSNR and SSIM values. Bilateral Filtering outperforms the other two methods, achieving the highest PSNR values, indicating superior noise reduction while preserving image details. Its SSIM stability is also the highest, demonstrating that it maintains structural similarities better than Gaussian or Median Filtering.

Gaussian Filtering, while effective in noise reduction, has moderate performance. It does not preserve edges as well as Bilateral or Median Filtering and exhibits fluctuations in SSIM values, suggesting potential loss of fine details.

Median Filtering provides a good balance between noise reduction and edge preservation. It performs better than Gaussian Filtering, particularly in reducing salt-and-pepper noise. However, while it offers high PSNR and SSIM values, it is still slightly less effective than Bilateral Filtering in overall performance.

Selecting the appropriate filtering technique depends on the specific application. If edge preservation and high noise reduction are crucial, Bilateral Filtering is the best option. For general smoothing, Gaussian Filtering is useful, but it may introduce blurring. Median Filtering is a suitable choice for removing specific types of noise while maintaining structural integrity.

Conclusion:

- **Bilateral Filtering** is the most effective noise reduction technique, offering high PSNR and stable SSIM.
- **Median Filtering** is a good alternative, performing better than Gaussian Filtering.
- **Gaussian Filtering** provides moderate performance but can blur image edges.

Understanding these differences helps in selecting the most suitable filtering technique for specific image processing applications.

4.2 Color Space Conversion: Noise reduction is an essential aspect of image processing, aimed at minimizing unwanted distortions while preserving crucial image details. Noise in images can result from various factors such as environmental conditions, sensor limitations, and transmission errors. Removing noise effectively enhances image quality, improving its usability for further analysis and applications such as medical imaging, computer vision, and surveillance.

There are multiple approaches to noise reduction, each with its strengths and limitations. This document discusses two widely used methods: Grayscale

Transformation and HSV Color Space Conversion, comparing their effectiveness based on PSNR (Peak Signal-to-Noise Ratio) and SSIM (Structural Similarity Index).

Color space Conversion techniques :

A. Grayscale Transformation

- Converts the image to a single-channel representation, reducing computational complexity.
- Emphasizes intensity variations, which can help in better noise reduction.
- Commonly used in edge detection and feature extraction.
- Shows stability in SSIM values, indicating strong structural similarity preservation.

B. HSV Color Space Conversion

- Separates image components into Hue, Saturation, and Value, allowing better handling of color information.
- Useful in applications requiring color-based segmentation and feature recognition.
- Provides a broader PSNR range but shows fluctuations in SSIM values.
- Helps retain color fidelity while reducing noise in specific scenarios.

Comparative Analysis:

Metric	**Grayscale Images** (PSNR & SSIM)	**HSV Color Space Conversion** (PSNR & SSIM)

PSNR Values	Ranges from ~8 to 13 dB	Ranges from ~7 to 17 dB
SSIM Values	Ranges from ~0.05 to 1	Ranges from ~0.1 to 0.7
Trend Analysis	Grayscale PSNR shows fluctuation with a notable dip	HSV PSNR fluctuates with a sharp increase in later images
SSIM Trend	Sharp increase to 1, maintaining stability	Gradual increase, stabilizing around 0.7
Overall Quality	Grayscale transformation maintains high similarity	HSV has varying PSNR and lower SSIM scores

The comparison between grayscale image processing and HSV color space conversion highlights significant differences in noise reduction effectiveness:

- **PSNR (Peak Signal-to-Noise Ratio):** Grayscale images exhibit a PSNR range of approximately 8 to 13 dB, showing fluctuations with occasional dips. HSV color space conversion, on the other hand, covers a broader PSNR range of about 7 to 17 dB, with a noticeable sharp increase in later images. This suggests that while HSV conversion can sometimes outperform grayscale in noise reduction, it may also introduce inconsistencies.

- **SSIM (Structural Similarity Index):** Grayscale images demonstrate a steady increase in SSIM values, reaching close to 1 and maintaining stability, signifying excellent structural preservation. In contrast, HSV transformation presents a gradual rise in SSIM values but stabilizes around 0.7, indicating relatively lower structural integrity retention compared to grayscale.

- **Overall Quality:** Grayscale processing provides more consistent image quality due to its stable SSIM values and well-maintained structural details. Meanwhile, HSV color space conversion shows variations in quality, offering higher PSNR improvements in certain cases but at the cost of lower SSIM stability.

Conclusion:

- **Grayscale Transformation** is effective for preserving structural similarity and producing consistent noise reduction results.
- **HSV Color Space Conversion** offers a wider range of noise reduction performance but exhibits variability in structural preservation.
- The choice between the two depends on the specific requirements of an application—grayscale for stable structural retention and HSV for color-sensitive applications with potential noise reduction benefits.

Understanding these differences is crucial in selecting the appropriate technique based on the desired balance between noise reduction effectiveness and structural similarity maintenance.

4.3 Contrast Enhancement:

Contrast enhancement is a fundamental image processing technique used to improve the visibility of details in images. It works by increasing the difference between the light and dark areas of an image, making objects more distinguishable. This is particularly useful in medical imaging, remote sensing, and low-light photography, where image clarity is crucial.

Different contrast enhancement techniques offer varying levels of improvement depending on the image characteristics and application requirements. This document focuses on three methods: CLAHE (Contrast Limited Adaptive Histogram Equalization), Gamma Correction, and Histogram Equalization, analyzing their effectiveness using PSNR (Peak Signal-to-Noise Ratio) and SSIM (Structural Similarity Index).

Contrast Enhancement Techniques:

A. CLAHE (Contrast Limited Adaptive Histogram Equalization)

- Enhances local contrast by applying histogram equalization to small image regions.
- Helps in preserving details while preventing over-enhancement in uniform areas.
- Maintains a high consistency in image quality.
- Achieves higher SSIM (0.55 - 0.9) and PSNR (13 - 21 dB) compared to other methods.

B. Gamma Correction

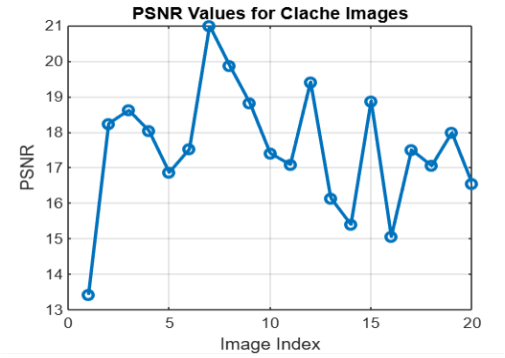
- Adjusts image brightness by applying a power-law transformation.
- Useful for correcting underexposed or overexposed images.
- Provides moderate consistency in enhancement results.
- Achieves SSIM values between 0.4 - 0.75 and PSNR in the range of 12 - 19 dB.

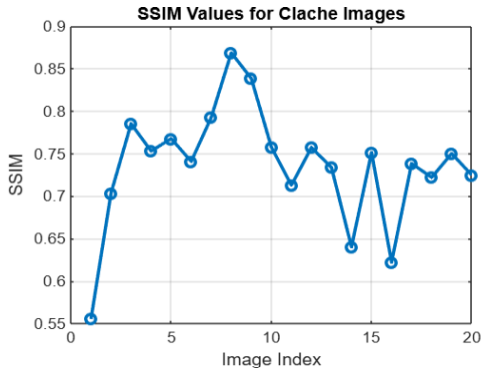
C. Histogram Equalization

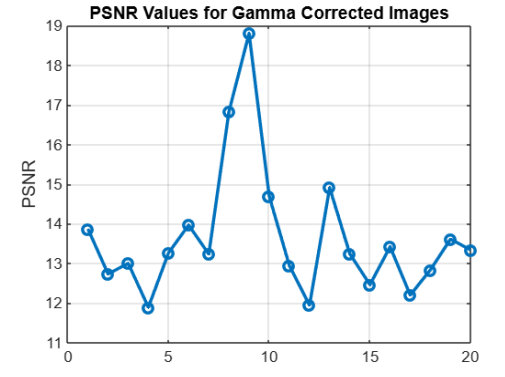
- Improves overall contrast by redistributing pixel intensity values.
- Suitable for images with low contrast but may introduce noise and artifacts.
- Exhibits low consistency with a wide range of results.
- Achieves SSIM between 0.3 - 0.8 and PSNR from 6 - 16 dB.

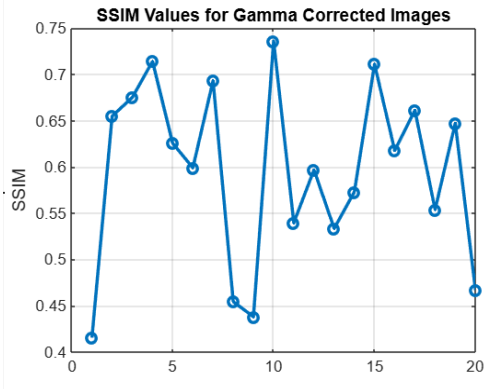
Comparative Analysis:

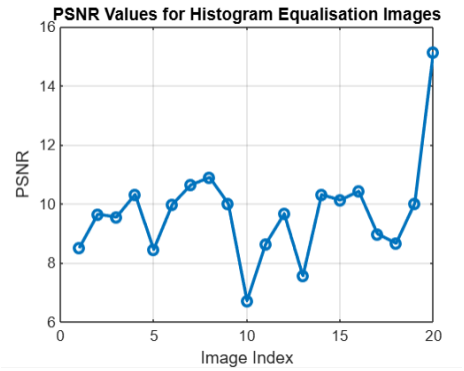
Metric	CLAHE	Gamma Correction	Histogram Equalisation
PSNR Range	13 - 21	12 - 19	6 - 16
PSNR Peak	Around Image 5 and Image 10	Around Image 10	At Image 20
SSIM Range	0.55 - 0.9	0.4 - 0.75	0.3 - 0.8
SSIM Peak	Around Image 5 and Image 8	Around Image 5 and Image 11	At Image 20
Consistency	High	Moderate	Low

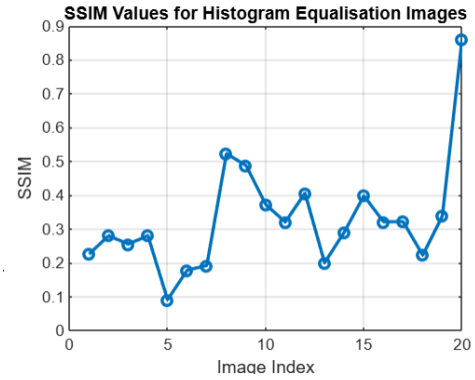












The comparative analysis highlights the strengths and weaknesses of the three contrast enhancement methods:

- **PSNR (Peak Signal-to-Noise Ratio):**
  - CLAHE provides the highest PSNR range (13 - 21 dB) with peaks around Image 5 and Image 10.
  - Gamma Correction follows with a PSNR range of 12 - 19 dB, peaking around Image 10.
  - Histogram Equalization has the lowest PSNR range (6 - 16 dB), peaking at Image 20.
- **SSIM (Structural Similarity Index):**
  - CLAHE achieves the highest SSIM range (0.55 - 0.9) with peaks around Image 5 and Image 8.
  - Gamma Correction has SSIM values between 0.4 - 0.75, peaking at Image 5 and Image 11.
  - Histogram Equalization has the lowest SSIM (0.3 - 0.8) with a peak at Image 20.
- **Overall Consistency:**
  - CLAHE is the most stable technique, maintaining high contrast enhancement with n minimal artifacts.

- Gamma Correction provides moderate consistency, effective in certain lighting conditions but sensitive to extreme adjustments.
- Histogram Equalization is the least consistent, as it can cause over-enhancement or noise in some cases.

Conclusion:

- CLAHE is the most effective technique for contrast enhancement, offering high consistency and well-preserved details.
- Gamma Correction is useful for controlled brightness adjustments but does not perform as well in high-contrast situations.
- Histogram Equalization can provide noticeable enhancement but may introduce artifacts and inconsistencies.
- The choice of method depends on the application—CLAHE for stable and detailed enhancement, Gamma Correction for brightness adjustments, and Histogram Equalization for improving low-contrast images.

4.4 Edge Detection: Edge detection is a crucial technique in image processing that identifies significant variations in pixel intensity, highlighting the boundaries of objects within an image. This is widely used in computer vision, medical imaging, object detection, and feature extraction. By emphasizing edges, edge detection techniques help in analyzing shapes, structures, and textures while reducing unnecessary details.

Different edge detection methods vary in accuracy, noise sensitivity, and performance. This document focuses on three well-known techniques: Canny,

Laplacian of Gaussian (Log), and Sobel, comparing their effectiveness using PSNR (Peak Signal-to-Noise Ratio) and SSIM (Structural Similarity Index).

Edge Detection Techniques:

A. Canny Edge Detection:

- A multi-step algorithm that smooths the image, detects intensity gradients, applies non-maximum suppression, and uses double thresholding.
- Produces well-defined edges with moderate consistency.
- Provides PSNR values between 4 - 7.5 dB and SSIM values ranging from -0.005 to 0.02.

B. Laplacian of Gaussian (LoG):

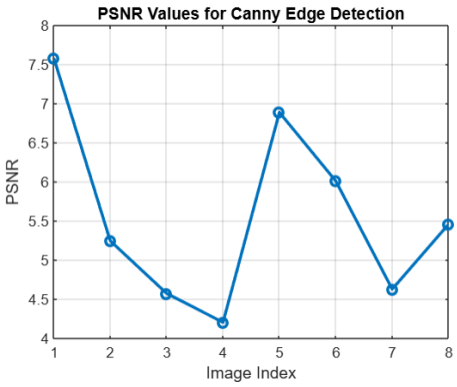
- Uses a combination of Gaussian smoothing and the Laplacian operator to detect edges.
- More sensitive to noise, leading to lower consistency in results.
- Achieves PSNR values between 4 - 9 dB and SSIM values ranging from -0.25 to -0.02.

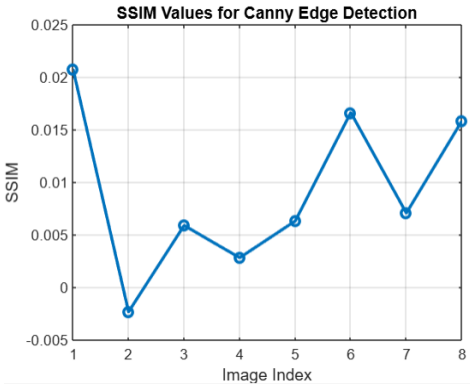
C. Sobel Edge Detection:

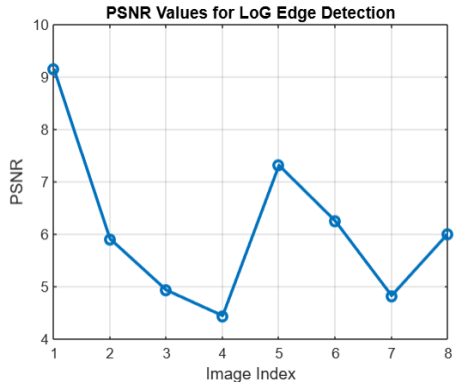
- Computes gradients in horizontal and vertical directions to detect edges.
- Provides higher consistency and smooth edge transitions.
- Exhibits PSNR values between 5 - 10.5 dB and SSIM values ranging from 0.02 to 0.2.

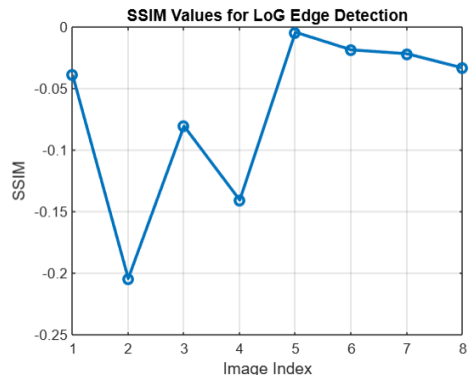
Comparative Analysis:

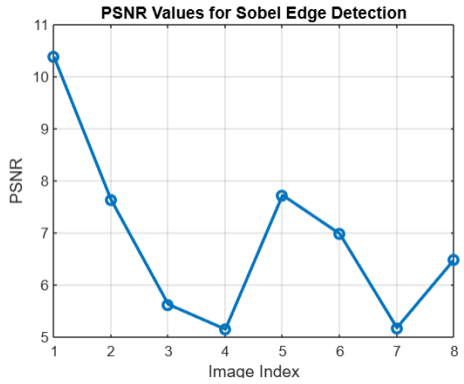
Metric	Canny	LoG	Sobel
PSNR Range	4 - 7.5	4 - 9	5 - 10.5
PSNR Peak	Around Image 1 and Image 5	Around Image 1 and Image 5	Around Image 1 and Image 5
SSIM Range	-0.005 - 0.02	-0.25 - -0.02	0.02 - 0.2
SSIM Peak	Around Image 1 and Image 5	Around Image 5	Around Image 1 and Image 5
Consistency	Moderate	Low	High

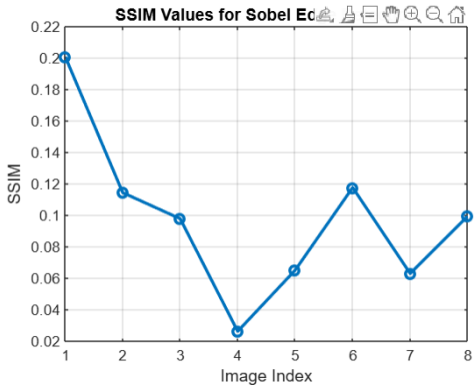












The comparative analysis of these edge detection techniques highlights their performance in terms of noise resistance, clarity, and consistency:

- **PSNR (Peak Signal-to-Noise Ratio):**
  - Canny shows PSNR values between 4 - 7.5 dB, peaking around Image 1 and Image 5.
  - LoG has a broader PSNR range of 4 - 9 dB, with peaks at Image 1 and Image 5.
  - Sobel exhibits the highest PSNR range of 5 - 10.5 dB, peaking at Image 1 and Image 5.
- **SSIM (Structural Similarity Index):**
  - Canny produces SSIM values between -0.005 and 0.02, peaking at Image 1 and Image 5.
  - LoG has the lowest SSIM range of -0.25 to -0.02, peaking at Image 5.
  - Sobel provides the highest SSIM range of 0.02 to 0.2, peaking at Image 1 and Image 5.
- **Overall Consistency:**
  - Canny provides moderate consistency, balancing noise reduction and edge sharpness.

- LoG shows low consistency, making it less reliable due to its sensitivity to noise.
- Sobel delivers high consistency, making it more suitable for stable edge detection.

Conclusion:

- Canny Edge Detection offers a balance between noise resistance and edge clarity, making it a preferred choice for general applications.
- LoG Edge Detection is sensitive to noise and may introduce artifacts, leading to lower consistency.
- Sobel Edge Detection provides the most consistent performance, making it effective for well-structured edges.
- The choice of method depends on the application—Canny for balanced results, LoG for high-detail extraction, and Sobel for stable and smooth edge transitions.

4.5 Segmentation:

Segmentation is a fundamental technique used in various domains like machine learning, computer vision, networking, and data processing. It involves partitioning a dataset, image, or market into distinct subsets based on predefined criteria, enhancing efficiency, accuracy, and targeted decision-making.

Segmentation Techniques:

A. Thresholding-based Methods

- Best for:
  Images with high contrast between foreground and background.
- Simple tasks like separating text from a plain background.

- Limitations:
  Doesn't work well with complex images or varying lighting conditions.
- Recommended technique: Otsu's method for automatic threshold selection.

B. Watershed Segmentation

- Best for:
  Images with overlapping objects, like cells in microscopy images.
- Limitations:
  Over-segmentation is common, may require preprocessing (e.g., distance transform).
- Recommended technique: Marker-controlled watershed to reduce over-segmentation.

C. Region Growing Segmentation

- Region growing is a specific technique under the umbrella of region-based segmentation. It starts with one or more seed points and iteratively adds neighboring pixels that are similar to the region based on a predefined criterion (e.g., intensity difference below a threshold).
- **Steps in Region Growing:**
  1. Initialize seed points (manually or automatically).
  2. Compare each unallocated neighboring pixel to the region.
  3. Add the pixel to the region if it meets the similarity criterion.
  4. Repeat until no more pixels can be added.
- **Advantages:**

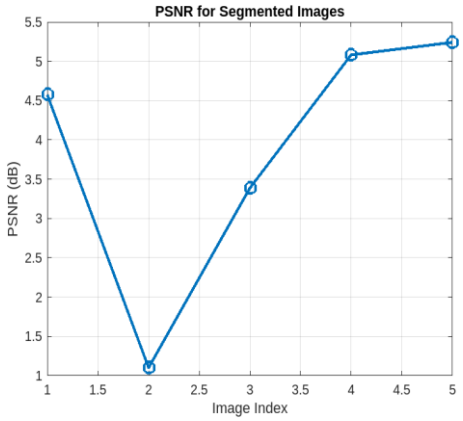
a. Works well for images with connected regions of similar intensity.
b. Simple to implement and interpret.

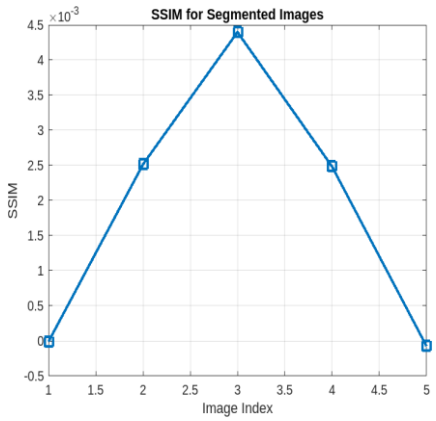
- **Limitations:**

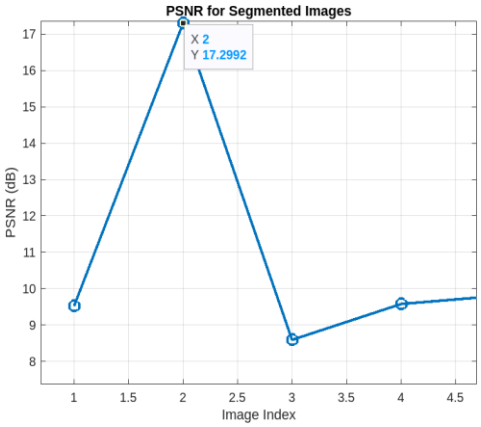
a. Sensitive to the choice of seed points.
b. May result in over-segmentation if the similarity criterion is not well-defined.

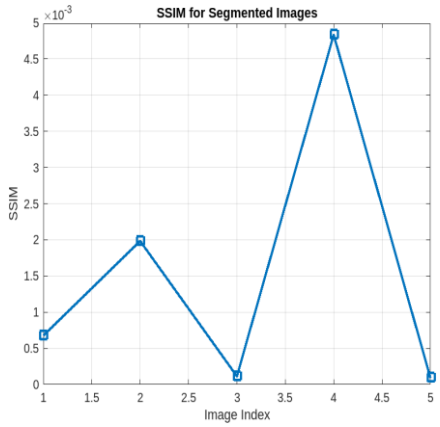
Comparative Analysis:

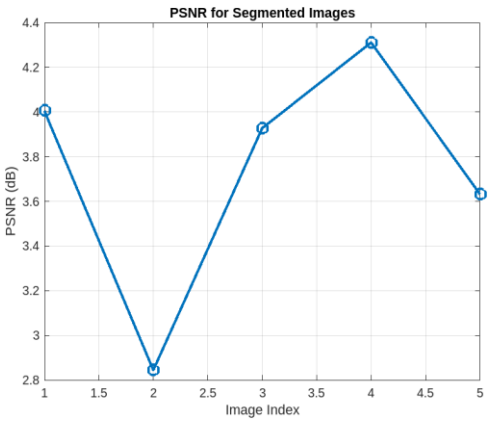
Graph Name	Metric Type	Observation	Trend
w1	PSNR	Peak at index 4 (~4.3 dB), lowest at index 2 (~2.8 dB)	V-shaped fluctuation
w2	SSIM	Peak at index 2 (~4.7×10^{-3}), lowest at index 1 and 5 (~0)	Sharp rise and fall
t1	PSNR	Peak at index 2 (~17.3 dB), lowest at index 3 (~8.5 dB)	Steep rise and drop
t2	SSIM	Peak at index 4 (~4.5×10^{-3}), lowest at index 3 and 5 (~0)	Increasing till 4, then dropping
r1	PSNR	Peak at index 5 (~5.2 dB), lowest at index 2 (~1.0 dB)	U-shaped fluctuation
r2	SSIM	Peak at index 3 (~4.5×10^{-3}), lowest at index 1 and 5 (~0)	Symmetric rise and fall

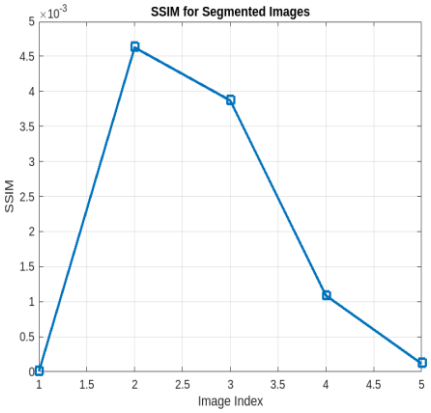












4.6 Object Detection:

Object detection is a crucial task in computer vision that involves identifying and locating objects in images or videos. It extends beyond simple image classification by not only recognizing the presence of an object but also determining its position within the frame. Object detection is widely used in applications such as surveillance, autonomous driving, and medical imaging.

Various techniques have been developed for object detection, including traditional methods and deep learning-based approaches. Traditional methods rely on handcrafted features, while deep learning models use convolutional neural networks (CNNs) to automatically learn features from data. This document explores and compares different object detection techniques based on their accuracy, performance, and computational requirements.

Object Detection Techniques:

Several methods are used for object detection, each with its advantages and limitations:

A. You Only Look Once (YOLO):

- A single-shot detector that divides an image into grids and predicts bounding boxes and class probabilities in one pass.
- Offers fast inference speeds, making it ideal for real-time applications.

B. Region-Based Convolutional Neural Networks (RCNN)

(Placeholder for RCNN details)

Comparative Analysis:

This section compares different object detection models based on training and validation losses, precision, recall, and overall stability. The table below presents a comparative analysis of different object detection models.

YOLO, R-CNN and U-Net:

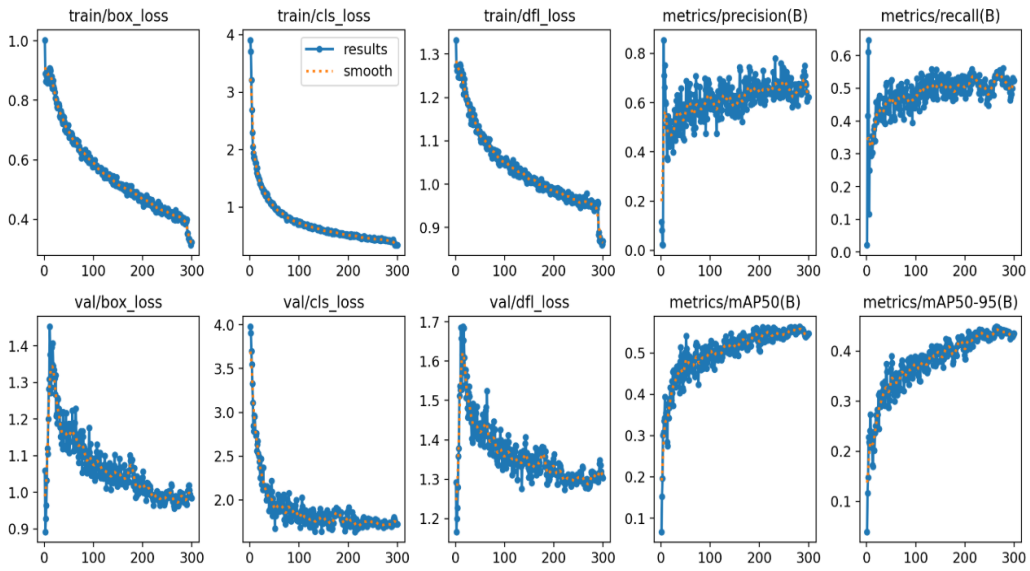
Metric	**YOLO**	**R-CNN**	**U-Net**
Primary Application	Real-time object detection	High-accuracy object detection	Image segmentation (pixel-wise classification)
Architecture	Single-stage, end-to-end network that directly predicts bounding boxes and classes	Two-stage process with region proposals followed by classification	Encoder-decoder network with skip connections for detailed segmentation
Average Precision	Shows a balanced	Often exhibits higher precision	Precision is evaluated at the

	performance – generally moderate to high precision; effective for larger objects	due to the refined region proposals, especially for small or cluttered objects	pixel level; performance depends on accurate delineation of object boundaries
Average Recall	Provides good recall overall, though may miss very small objects under certain conditions	Typically achieves higher recall by capturing a broader range of object proposals	Recall is tuned for segmentation tasks and usually maintains consistency across varied object sizes
Precision-Recall Curve	The curve is smooth and balanced, indicating consistent detection across varying thresholds	The curve tends to be steeper with a larger area under the curve, reflecting more robust detection	The curve is adapted for pixel-level evaluation, showing the trade-off between accurate boundaries and noise reduction
Training Loss Curve	Often shows a faster and smoother convergence with relatively lower fluctuations in training loss	Can exhibit more fluctuations and a slower convergence rate because of the multi-stage training process	Generally demonstrates a steady decline, reflecting consistent improvement over epochs
Inference Speed	Very fast – optimized for	Slower – the two-stage approach is more	Moderate – optimized for segmentation

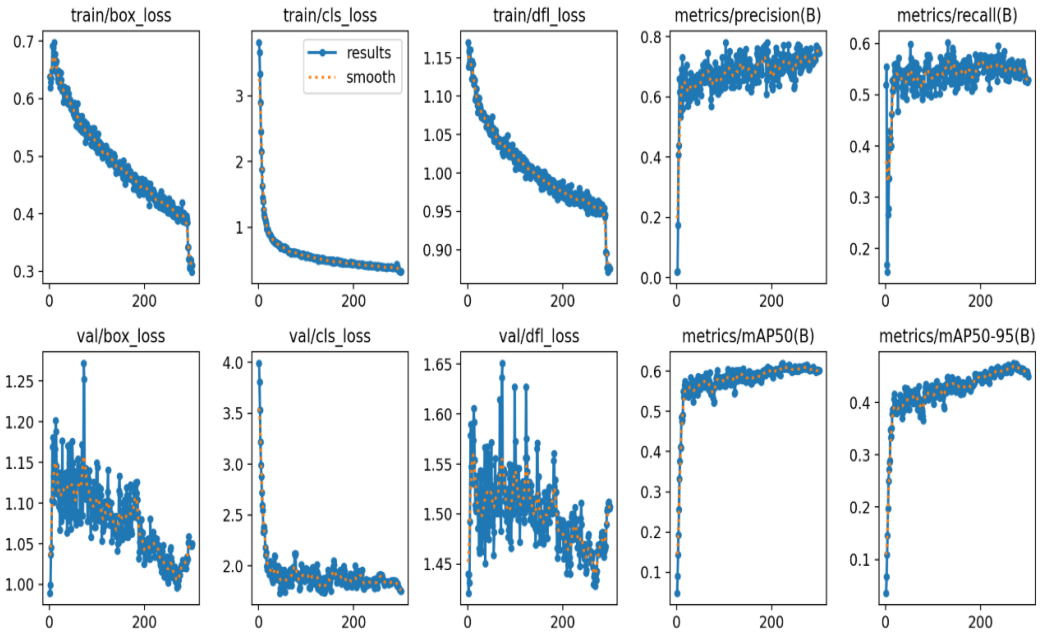
	real-time applications	computationally intensive	tasks, though not typically used where real-time detection is critical
Computational Complexity	Lower complexity with fewer parameters, making it suitable for edge devices and real-time applications	Higher complexity and computational load due to the region proposal and classification stages	Moderate complexity, balancing between detailed output and computational efficiency

YOLO:

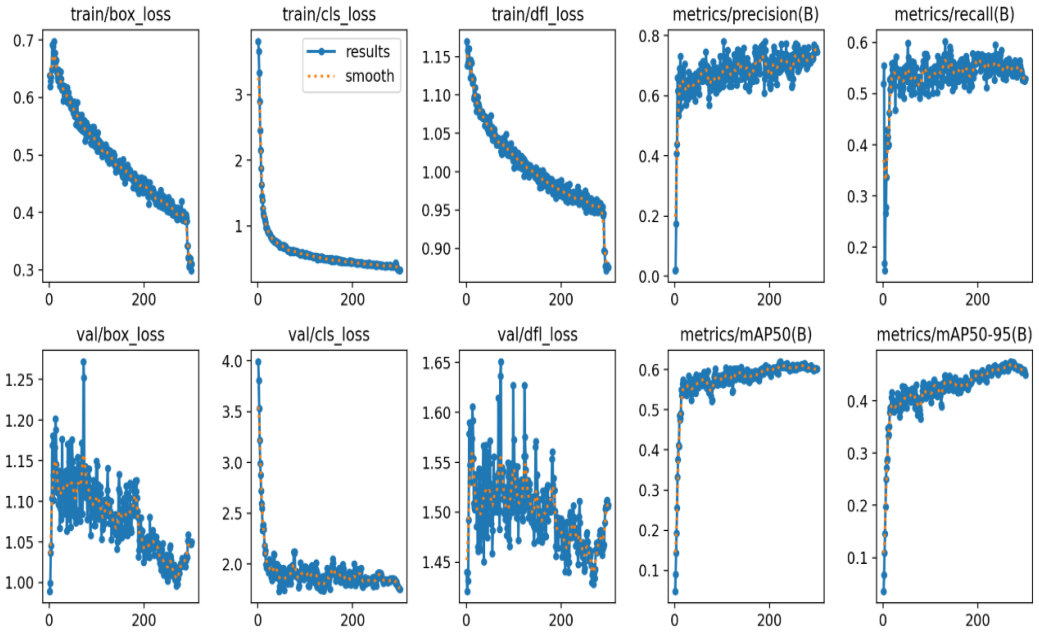
Model1(v4)



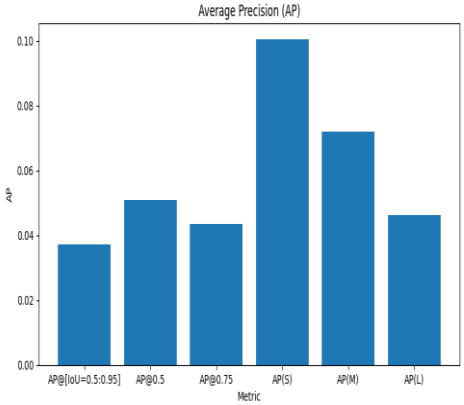
Model 2 (v6)

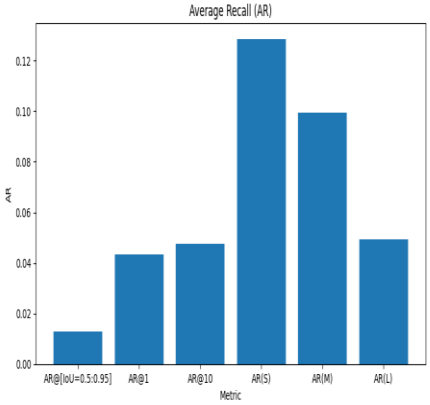


Model 3 (v8)

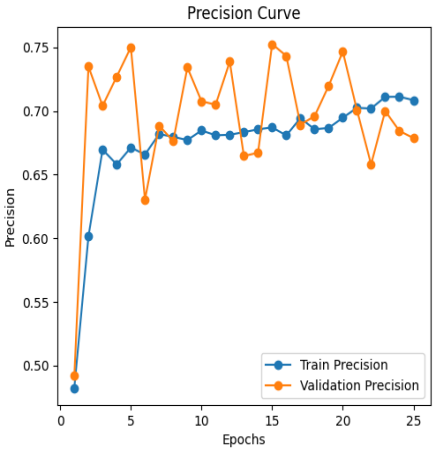


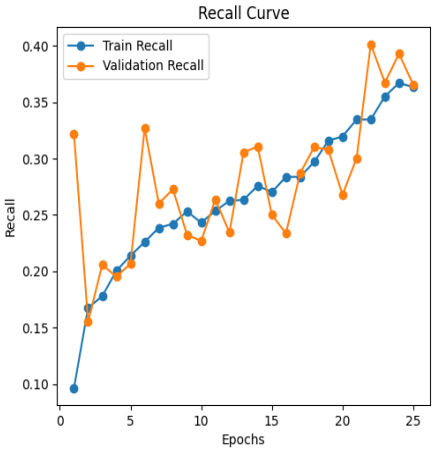
- **R-CNN:**

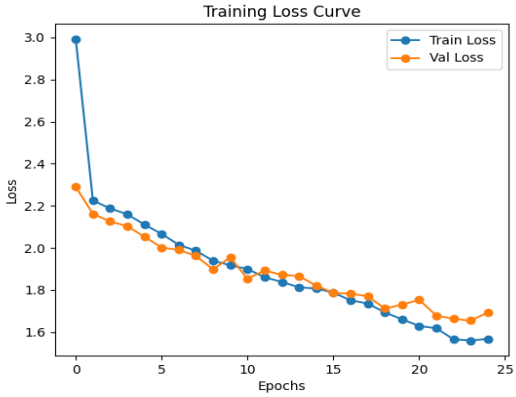

Average Precision (AP)
AP
Metric
AP@[IoU=0.5:0.95]
AP@0.5
AP@0.75
AP(S)
AP(M)
AP(L)


Average Recall (AR)
AR
Metric
AR@[IoU=0.5:0.95]
AR@1
AR@10
AR(S)
AR(M)
AR(L)

- **U-Net**


Precision Curve
Precision
Epochs
Train Precision
Validation Precision


Recall Curve
Recall
Epochs
Train Recall
Validation Recall


Training Loss Curve
Loss
Epochs
Train Loss
Val Loss

4.7 Distance Estimation:

This project enhances assistive vision for the visually impaired using real-time object detection and distance estimation. A YOLOv8 model detects objects, while distance is calculated via a monocular camera. Audio feedback in multiple languages informs users of the nearest object, improving spatial awareness and safety.

- **Distance Estimation Model**

  Based on Triangle Similarity:

  Distance = (Known Width × Focal Length) / Pixel Width

  Focal Length = (Pixel Width × Known Distance) / Known Width

  Known Width is the object's actual width; Pixel Width is from the image; Focal Length is calibrated with a reference object.

- **Implementation Steps**

  YOLOv8 detects objects and generates bounding boxes. Pixel width is derived from bounding box coordinates to estimate distance. The nearest object is identified, and the system gives real-time audio feedback (e.g., "Chair detected at 50 centimeters").

- **Accuracy Optimization**

  Fixed object sizes can reduce accuracy. Improvements include using a pre-trained model with a size database or monocular depth estimation. Camera angle and perspective distortions can be addressed using stereo vision or LiDAR in future versions.

- **Low-Light & Edge Detection Enhancements**

  CLAHE enhances image contrast for low-light object detection. Canny Edge Detection identifies sharp edges (e.g., stairs), and edge width is used for distance calculation (e.g., "Sharp edge detected at 30 centimeters").

- **Future Enhancements**

  Plans include stereo cameras, LiDAR, smart glasses integration, haptic feedback, and spatial audio for directional awareness, offering a richer and safer navigation experience.

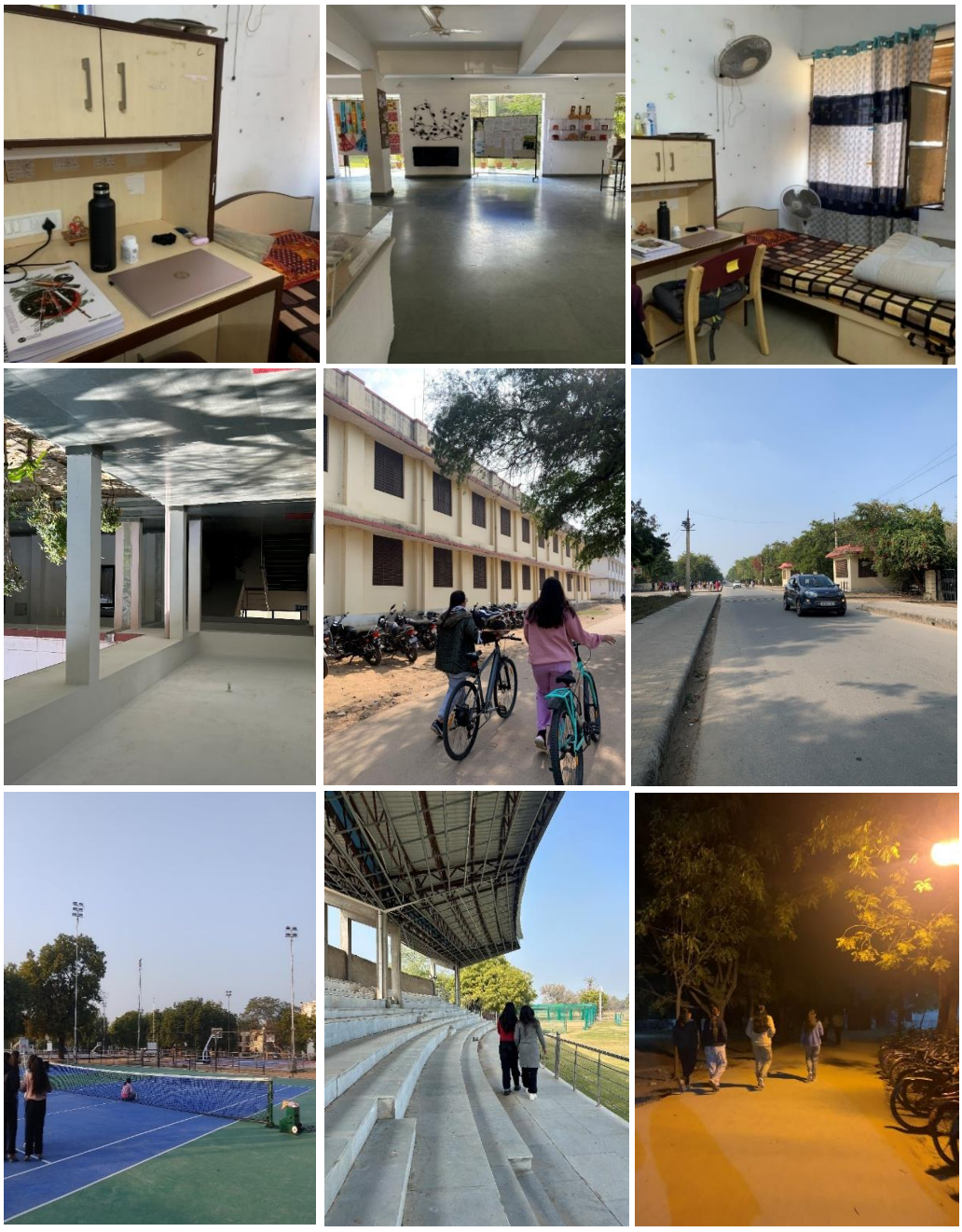
5. Conclusion

This work is a major contribution towards empowering blind and visually impaired persons by leveraging current state-of-the-art image processing, object detection, segmentation, and distance estimation techniques. The work solves imperative challenges like real-time performance, computational complexity, environmental variations, and system scalability, all necessary for developing usable assistive systems. Through the development of secure, AI-based systems that can recognize complicated, dynamic environments, the project seeks to give users precise and timely feedback that enables secure and self-assured navigation.

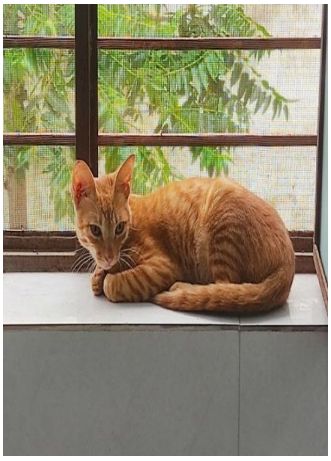
In addition, the focus on creating user-friendly interfaces guarantees that such technologies are made available to users who are not technically inclined, lowering the entry barrier and making it possible for more independence in daily life. The addition of unsupervised and weakly-supervised learning methods also decreases the dependency on expensive labeled datasets, accelerating the creation of adaptive systems that can be rolled out to a vast array of environments and applications.

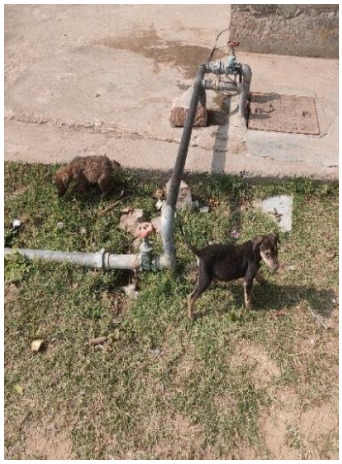
Finally, this work hopes to fill the chasm between technological advancement and human-centric design, providing a practical, scalable, and accessible solution that improves mobility, situation awareness, and quality of life for the visually impaired. It provides the groundwork for subsequent systems that not only support but empower users to engage with the world more freely, safely, and securely.

6. Appendices – Some sample datasets











7. References

1) He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).
2) Hamuda, E., Glavin, M., & Jones, E. (2016). A survey of image processing techniques for plant extraction and segmentation in the field. *Computers and electronics in agriculture*, *125*, 184-199.
3) Jamieson, L. H., Delp, E. J., Wang, C. C., Li, J., & Weil, F. J. (1992). A software environment for parallel computer vision. Computer, 25(2), 73-77.
4) Elad, M., Figueiredo, M. A., & Ma, Y. (2010). On the role of sparse and redundant representations in image processing. *Proceedings of the IEEE*, *98*(6), 972-982.
5) Abdulateef, S. K., & Salman, M. D. (2021). A Comprehensive Review of Image Segmentation Techniques. *Iraqi Journal for Electrical & Electronic Engineering*, *17*(2).
6) Minaee, S., Boykov, Y., Porikli, F., Plaza, A., Kehtarnavaz, N., & Terzopoulos, D. (2021). Image segmentation using deep learning: A survey. IEEE transactions on pattern analysis and machine intelligence, 44(7), 3523-3542
7) KKa, P., Ab, K., Ya, K., LMc, B., & DKa, S. (2015). Image processing tools and techniques used in computer vision for quality assessment of food products: a review. Int. J, 1, 1-16.
8) Cho, T. H., Conners, R. W., & Araman, P. A. (1990, June). A computer vision system for analyzing images of rough hardwood lumber. In [1990] Proceedings. 10th International Conference on Pattern Recognition (Vol. 1, pp. 726-728). IEEE.
9) Voulodimos, A., Doulamis, N., Doulamis, A., & Protopapadakis, E. (2018). Deep learning for computer vision: A brief review. Computational intelligence and neuroscience, 2018(1), 7068349.
10) Li, C., Xu, C., Gui, C., & Fox, M. D. (2010). Distance regularized level set evolution and its application to image segmentation. IEEE transactions on image processing, 19(12), 3243-3254.
11) Liu, X., Deng, Z., & Yang, Y. (2019). Recent progress in semantic image segmentation. Artificial Intelligence Review, 52, 1089-1106.

12) Vincent, L., & Dougherty, E. R. (2020). Morphological segmentation for textures and particles. In Digital image processing methods (pp. 43-102). CRC Press.
13) Yu, Y., Wang, C., Fu, Q., Kou, R., Huang, F., Yang, B., ... & Gao, M. (2023). Techniques and challenges of image segmentation: A review. Electronics, 12(5), 1199.
14) Wang, R., Lei, T., Cui, R., Zhang, B., Meng, H., & Nandi, A. K. (2022). Medical image segmentation using deep learning: A survey. IET image processing, 16(5), 1243-1267.
15) Hoeser, T., & Kuenzer, C. (2020). Object detection and image segmentation with deep learning on earth observation data: A review-part i: Evolution and recent trends. Remote Sensing, 12(10), 1667.
16) Dai, X., Yuan, X., & Wei, X. (2021). TIRNet: Object detection in thermal infrared images for autonomous driving. Applied Intelligence, 51(3), 1244-1261.
17) Zhang, M., Zhang, R., Yang, Y., Bai, H., Zhang, J., & Guo, J. (2022). ISNet: Shape matters for infrared small target detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 877-886).
18) Li, S., Li, Y., Li, Y., Li, M., & Xu, X. (2021). Yolo-firi: Improved yolov5 for infrared image object detection. *IEEE access*, *9*, 141861-141875.
19) Suo, J., Wang, T., Zhang, X., Chen, H., Zhou, W., & Shi, W. (2023). HIT-UAV: A high-altitude infrared thermal dataset for Unmanned Aerial Vehicle-based object detection. *Scientific Data*, *10*(1), 227.
20) Li, D., Salonidis, T., Desai, N. V., & Chuah, M. C. (2016, October). Deepcham: Collaborative edge-mediated adaptive deep learning for mobile object recognition. In 2016 IEEE/ACM Symposium on Edge Computing (SEC) (pp. 64-76). IEEE.
21) Liu, M., Ding, X., & Du, W. (2020, November). Continuous, real-time object detection on mobile devices without offloading. In 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS) (pp. 976-986). IEE

22) Gao, C., Meng, D., Yang, Y., Wang, Y., Zhou, X., & Hauptmann, A. G. (2013). Infrared patch-image model for small target detection in a single image. IEEE transactions on image processing, 22(12), 4996-5009.

23) Wu, X., Hong, D., & Chanussot, J. (2022). UIU-Net: U-Net in U-Net for infrared small object detection. IEEE Transactions on Image Processing, 32, 364-376.

24) Li, Y., Li, S., Du, H., Chen, L., Zhang, D., & Li, Y. (2020). YOLO-ACN: Focusing on small target and occluded object detection. IEEE access, 8, 227288-227303.

25) Kolev, K., Tanskanen, P., Speciale, P., & Pollefeys, M. (2014). Turning mobile phones into 3D scanners. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3946-3953).

26) Zhou, Y., Liu, Q., Zhu, H., Li, Y., Chang, S., & Guo, M. (2022). Mogde: Boosting mobile monocular 3d object detection with ground depth estimation. *Advances in Neural Information Processing Systems*, *35*, 2033-2045.

27) Ming, Y., Meng, X., Fan, C., & Yu, H. (2021). Deep learning for monocular depth estimation: A review. *Neurocomputing*, *438*, 14-33.

8.List of Publication(s)

- **6th International Conference on Deep Learning, Artificial Intelligence and Robotics (ICDLAIR) Organized by Department of Computer Engineering National Institute of Technology, Kurukshetra 6-8 Dec 2024**

  Presented a paper titled "A review on Standard Technique used in Object Detection from a Distance" in ICDLAIR 2024 and received Certification of Appreciation.