# AI Powered Smart Glasses for Visually Impaired

*A Project Report*

*Submitted to the APJ Abdul Kalam Technological University*

*in partial fulfillment of requirements for the award of degree*

***Bachelor of Technology***

*in*

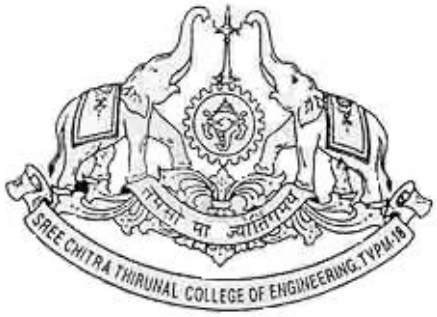
***Computer Science and Engineering***

*by*

**Athira D(SCT20CS023)**

**Prakash Roy(SCT20CS052)**

**S Gowrisankar(SCT20CS058)**




**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**SREE CHITRA THIRUNAL COLLEGE OF ENGINEERING THIRUVANANTHAPURAM**

**KERALA**

**May 2024**

**DEPT. OF COMPUTER SCIENCE AND ENGINEERING**

**SREE CHITRA THIRUNAL COLLEGE OF ENGINEERING**

**THIRUVANANTHAPURAM**

**2023-24**

## CERTIFICATE

This is to certify that the report entitled **AI Powered Smart Glasses for Visually Impaired** submitted by **Athira D** (SCT20CS023), **Prakash Roy** (SCT20CS052) & **S Gowrisankar** (SCT20CS058) to the APJ Abdul Kalam Technological University in partial fulfillment of the B.Tech. degree in Computer Science and Engineering is a bonafide record of the project work carried out by them under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

**Prof. Haripriya B S**
(Project Guide)
Assistant Professor
Dept. of CSE
SCT College of Engineering
Thiruvananthapuram

**Dr. Subusurendran**
(Project Coordinator)
Professor
Dept. of CSE
SCT College of Engineering
Thiruvananthapuram

**Dr. Chitharanjan Karat**
(Project Coordinator)
Professor
Dept. of CSE
SCT College of Engineering
Thiruvananthapuram

**Prof. Rejimol Robinson R R**
(Head of Department)
Associate Professor
Dept. of CSE
SCT College of Engineering
Thiruvananthapuram

**External Supervisor**

## DECLARATION

We hereby declare that the project report **AI Powered Smart Glasses for Visually Impaired**, submitted for partial fulfillment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by us under supervision of Prof. Haripriya B S

This submission represents our ideas in our own words and where ideas or words of others have been included, we have adequately and accurately cited and referenced the original sources.

We also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

**Athira D**

**Prakash Roy**

**S Gowrisankar**

Thiruvananthapuram

21-04-2024

# Acknowledgement

We take this opportunity to express my deepest sense of gratitude and sincere thanks to everyone who helped us to complete this work successfully. We express our sincere thanks to **Prof. Rejimol Robinson**, Head of Department, Computer Science and Engineering, Sree Chitra Thirunal College of Engineering for providing us with all the necessary facilities and support.

We would like to express my sincere gratitude to the **Dr. Chitharanjan Karat** and **Dr. Subu Surendran**, Department of Computer Science and Engineering, Sree Chitra Thirunal College of Engineering Thiruvananthapuram for the support and co-operation.

We would like to place on record my sincere gratitude to our project guide **Prof. Haripriya B S**, Assistant Professor, Department of Computer Science and Engineering, Sree Chitra Thirunal College of Engineering for the guidance and mentorship throughout this work.

Finally we thank our family, and friends who contributed to the succesful fulfilment of this project work.

**Athira D**

**Prakash Roy**

**S Gowrisankar**

# Abstract

Visually impaired individuals encounter significant challenges navigating and interacting with their surroundings. Simple tasks, such as safely traversing through crowded areas or identifying objects, pose considerable difficulties due to the lack of visual cues. Moreover, the inability to perceive spatial distances increases the risk of collisions and accidents, limiting their independence and overall quality of life. Traditional aids, while helpful, often fall short in providing comprehensive solutions, leaving gaps in essential functionalities like real-time obstacle detection and efficient access to information.

The system aims to alleviate these challenges by seamlessly integrating cutting-edge technologies. Utilizing stereo cameras and the StereoBM algorithm, the system accurately estimate distances to detect obstacles, facilitating a voice alert system for user safety. The intuitive voice command interface empowers users to access a range of functionalities seamlessly, from object detection to image captioning and text recognition, providing invaluable assistance in daily tasks and enhancing overall accessibility and independence. Through the integration of edge AI and innovative computing, our system endeavors to revolutionize the experience of visually impaired individuals, fostering greater autonomy and inclusion in a visually oriented world.

# Contents

# List of Figures

# Chapter 1

# Introduction

The project titled ”AI Powered Smart Glasses for Visually Impaired” represents a groundbreaking initiative at the intersection of cutting-edge technology and social impact. This transformative endeavor is driven by the mission to enhance the daily lives of individuals facing visual impairments by providing them with an innovative assistive device. At its technological core is the NVIDIA Jetson Nano, a powerful edge AI chip meticulously chosen to facilitate efficient model deployment. This project integrates a suite of hardware components, including a stereo camera, microphone, and earphone strategically combined to create a comprehensive solution.

The smart glasses operate on a robust software foundation that includes BLIP for real-time object recognition and image captioning, Tesseract OCR for text extraction, BERT for natural language generation, and a sophisticated voice command interface. Together, these components empower users with a range of functionalities, from real-time object identification to accurate distance estimation, all delivered through an accessible text-to-speech format. The overarching goal is to foster independence, accessibility, and inclusivity for visually impaired individuals, showcasing the potential of technology to address real-world challenges. This introduction sets the stage for a detailed exploration of the project's design, implementation, and its societal impact.

## 1.1 Purpose of the Project

The project endeavors to create smart glasses for visually impaired individuals, offering features like object recognition, text-to-speech, and voice commands. The goal is to provide a user-friendly and independent navigation solution, addressing the limitations of existing systems for the visually impaired community.

## 1.2 Intended Audience and Document Overview

The intended audience for this document is individuals with visual impairments who seek innovative solutions to enhance their daily lives. Additionally, developers, researchers, and stakeholders interested in the field of assistive technology, artificial intelligence, and human-computer interaction may find valuable insights in understanding the methodologies and technologies employed in this project.

This document serves as a comprehensive guide to the AI-powered smart glasses project designed for visually impaired individuals. It outlines the objectives, methodologies, and technologies employed in developing this assistive device. The document covers key aspects such as system architecture, and implementation details. For developers and researchers, it provides insights into the methodologies and algorithms used, contributing to the broader discourse on assistive technologies.

## 1.3 Scope of the Project

- Develop AI-powered smart glasses for visually impaired individuals.
- Utilize Nvidia Jetson Nano for edge AI processing.
- Incorporate stereo camera, microphone, and earphone for real-time data processing and feedback.
- Implement BLIP, Tesseract OCR, and StereoBM algorithms for object recognition, text-to-speech conversion, and obstacle detection.
- Prioritize accessibility and inclusivity throughout the design process, aiming for intuitive user interface and continuous user feedback for refinement.

# Chapter 2

# Literature Review

## 2.1 Existing Systems

### 2.1.1 Object Detection and Narrator for Visually Impaired People

This paper [1] presents a groundbreaking system aimed at revolutionizing the way visually impaired individuals interact with their environment. It offers a novel solution by integrating two crucial functionalities: real-time object detection and audio narration within a convenient and accessible platform – the user's own smartphone.

Leveraging the increasing capabilities of mobile technology, this system utilizes the phone's camera as the primary input device. Captured images are then processed through a sophisticated combination of technologies, including:

- TensorFlow: This powerful open-source machine learning framework handles the heavy lifting of object detection, enabling the system to identify a diverse range of objects in real-time.
- VGG16 Convolutional Neural Network (CNN) model: This pre-trained model acts as the backbone of the object detection process, recognizing various objects with high accuracy.

The system operates through a web interface, where captured images are transmitted to a server for processing. This approach ensures efficient utilization of resources and removes the need for computationally intensive tasks on the mobile device itself. Once

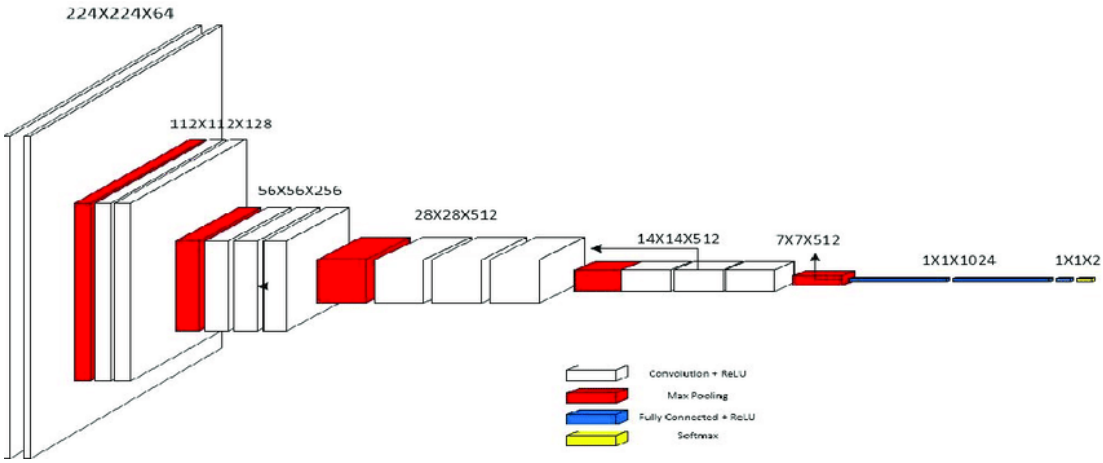



Figure 2.1: CNN Architecture based on VGG16

objects are identified, the system accesses a browser-based voice library to generate clear and concise audio narrations, providing the user with immediate information about their surroundings.

This innovative system addresses a critical challenge faced by the visually impaired community: the inability to access real-time visual information readily. By offering on-the-fly narration of detected objects, the system empowers users with greater independence and autonomy. They can now navigate their surroundings with confidence, making informed decisions based on their immediate environment.

The study's results highlight the efficacy of this approach. Extensive testing across various phone models and lighting conditions confirmed high accuracy in object detection, demonstrating the system's robustness and adaptability. This promising technology has the potential to significantly impact the lives of individuals with visual impairments, enhancing their accessibility and fostering greater independence in their daily activities.

### 2.1.2 Ultrasonic Sensor for Blind and Deaf Persons

Navigating the world presents unique challenges for individuals with combined sensory impairments, particularly those who are both blind and deaf. Existing assistive technologies often cater to only one sensory loss, leaving this population underserved and vulnerable to potential dangers in their surroundings. This research paper [2]aims to address this gap by proposing a novel device that combines two crucial

functionalities: ultrasonic obstacle detection and a dual-alert system.

The device utilizes an ultrasonic sensor capable of detecting obstacles within a 150 cm range. For blind users, this information is communicated through distinctive sound alerts that vary in intensity and frequency depending on the proximity and type of obstacle. This auditory feedback provides them with crucial awareness of their surroundings and helps them navigate safely. Deaf users benefit from a discreet vibration alert system activated by placing a finger on a designated button. This haptic feedback offers a clear and effective way for them to stay informed about potential hazards without relying on sound cues. By combining these two functionalities,

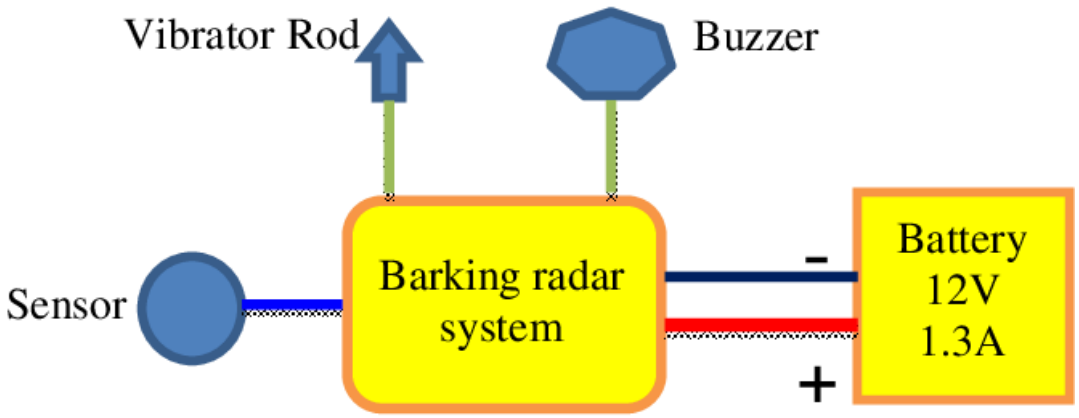



Figure 2.2: Blind and deaf device system block diagram

the device empowers individuals with combined sensory impairments to navigate their environment with greater independence and confidence. The real-time obstacle detection and dual-alert system significantly enhance their safety and quality of life, reducing their dependence on others and promoting greater freedom and mobility.

Further development efforts could focus on expanding the detection range, integrating with GPS and navigation systems for turn-by-turn guidance, and offering customizable alert preferences to cater to individual needs and preferences. These advancements have the potential to revolutionize the lives of individuals with combined sensory impairments by creating a more inclusive and accessible world where they can navigate their surroundings with confidence and participate fully in society.

### 2.1.3 Exploring Transformers in Natural Language Generation

The field of Natural Language Generation (NLG) has witnessed a seismic shift in recent years, driven by the emergence of attention mechanisms and Transformer models. Prior to this revolution, state-of-the-art NLG architectures, such as Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, were hampered by inherent limitations. The increasing length of sentences led to vanishing gradient problems, where the impact of earlier words on later ones became exponentially weaker, hindering their ability to capture long-range dependencies within the text. Additionally, the sequential nature of processing sentences word-by-word posed a significant bottleneck to parallelization, limiting their computational efficiency. The

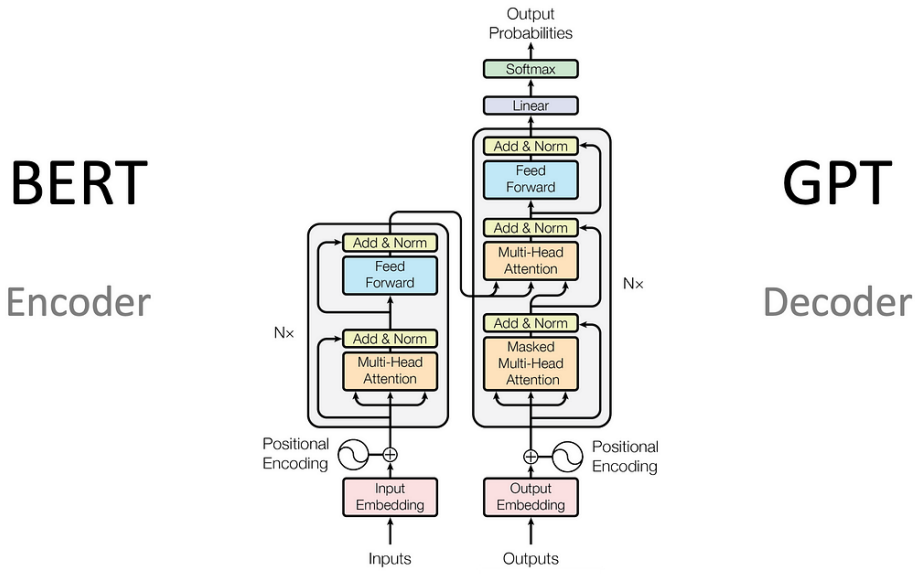



Figure 2.3: A Comprehensive Overview of Transformer-Based Models: Encoders, Decoders

arrival of Transformer models ushered in a new era for NLG, effectively addressing these limitations. Unlike previous architectures, Transformers leverage the power of attention mechanisms, enabling them to focus on relevant parts of the input sequence while processing the entire sentence at once. This eliminates the vanishing gradient problem and allows the model to capture complex relationships between words, regardless of their distance in the sentence. Furthermore, the inherent parallelization capabilities of Transformers significantly improve computational efficiency, making them ideal for handling large datasets and generating text on a massive scale.

This paper [3] delves deeper into the transformative impact of Transformers on NLG by exploring three major models: GPT, BERT, and XLNet. Each of these models offers unique advantages and holds significant potential for revolutionizing the field. GPT excels at text generation tasks, demonstrating remarkable ability to produce human-quality text, including poems, code, scripts, and even musical pieces. BERT, on the other hand, excels at pre-training, learning valuable representations of language that can be fine-tuned for diverse downstream tasks such as natural language inference and question answering. XLNet, building upon the strengths of both GPT and BERT, leverages a novel permutation language modeling objective that allows it to capture long-range dependencies and context more effectively, further pushing the boundaries of what's possible in NLG.

The rapid advancements in attention mechanisms, coupled with the transformative capabilities of Transformer models, have positioned NLG at the forefront of artificial intelligence research. From generating creative content and summarizing factual documents to facilitating natural human-computer interaction, the potential applications of NLG are vast and far-reaching. This paper provides a comprehensive exploration of this exciting new era in NLG, highlighting the transformative power of Transformers and paving the way for a future where language models play an increasingly vital role in our lives.

### 2.1.4 OCR by Open Source OCR Tool Tesseract

Optical Character Recognition (OCR) has become a crucial tool in today's digital world, transforming printed text into editable format with widespread applications. However, achieving accurate OCR results can be a challenge due to factors like text size, style, orientation, and complex backgrounds. This paper explores the capabilities of Tesseract, a popular open-source OCR tool, focusing on its history, architecture, and performance on diverse image types. Tesseract's architecture comprises several key components:

- Preprocessing: This stage prepares the input image for text recognition by applying techniques like noise reduction and binarization.

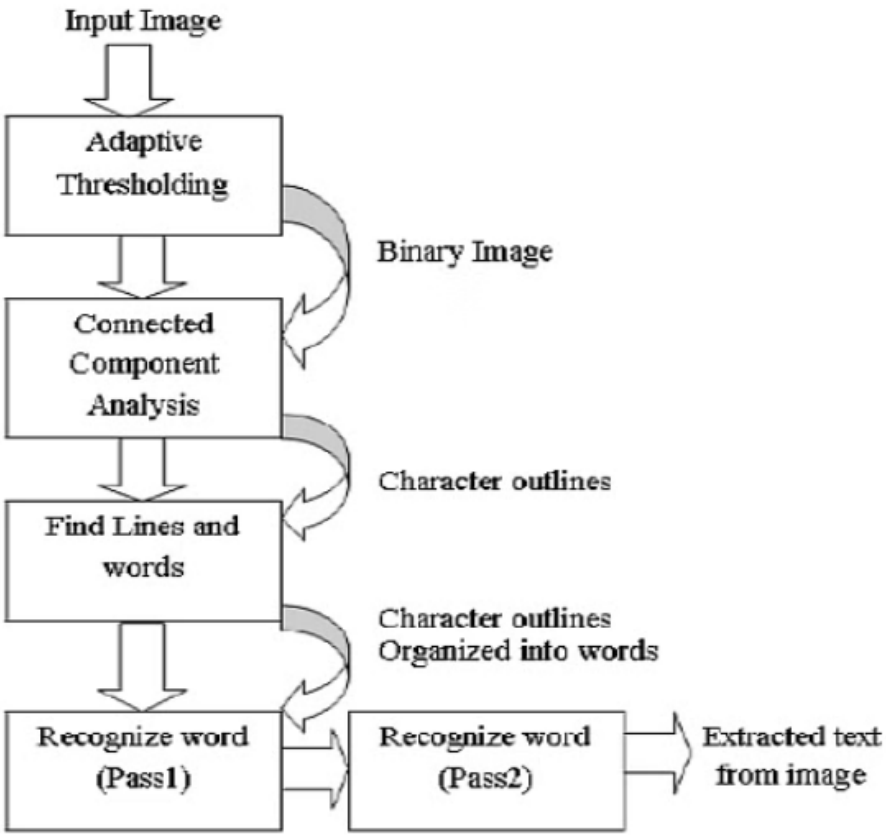



Figure 2.4: Architecture of Tesseract OCR

- Segmentation: The image is segmented into individual character regions for easier recognition.
- Feature extraction: Unique features are extracted from each character region.
- Character recognition: The extracted features are compared to a stored database of character patterns to identify the corresponding character.
- Post-processing: The recognized characters are combined to form the final output text.

The paper [4]explores Tesseract's performance on various image types, demonstrating its ability to handle complex scenarios. This includes images with different font styles, sizes, orientations, and backgrounds, highlighting the tool's robustness and adaptability. This case study of Tesseract demonstrates its undeniable impact on the OCR landscape. Its open-source nature, robust architecture, and impressive performance across diverse images solidify its position as a valuable tool for individuals and

organizations alike. The comparative analysis further underscores its competitiveness against commercial alternatives, making it a compelling choice for various OCR needs.

### 2.1.5 Performance Evaluation of Deep Learning Models on Embedded Platform

This research paper [5] focuses on the development and evaluation of Deep Learning models for traffic tracking and detection on resource-constrained embedded platforms. This technology is crucial for smart city initiatives, specifically in the realm of intelligent transportation. These sensors not only gather valuable traffic data, but also lessen the strain on communication networks by minimizing the amount of data transmitted and processed on centralized servers. This paradigm shift, known as Artificial Intelligence on the Edge (AIoT), emphasizes processing, storing, and extracting insights directly at the network's edge, before data is transferred to a central server.

The research aims to investigate, implement, and evaluate suitable machine learning models for deployment on embedded devices with limited computational resources. By leveraging computer vision and real-time object detection techniques, the research team set out to build an Edge-AI based traffic tracking and detecting sensor. Two popular models, MobileNet-SSD and YOLOv4, were chosen for in-depth study and implementation to compare their performance in vehicle counting and license plate detection applications. Furthermore, a novel approach utilizing the TensorRT engine was proposed to optimize these models and enhance their processing speed. The data used was collected from real-world traffic scenarios and parking lots in Vietnam. Over 11,700 images of Vietnamese vehicles and license plates were meticulously compiled and then used to train the models on the Google Colab platform. This ensured the models were trained on data relevant to the target application environment.

The performance evaluation revealed both models achieved high accuracy in real-time license plate detection and vehicle counting applications when implemented on the NVIDIA Jetson Nano embedded platform. Notably, the mean Average Precision (mAP) for both models exceeded 90 The MobileNet-SSD model demonstrated ex-

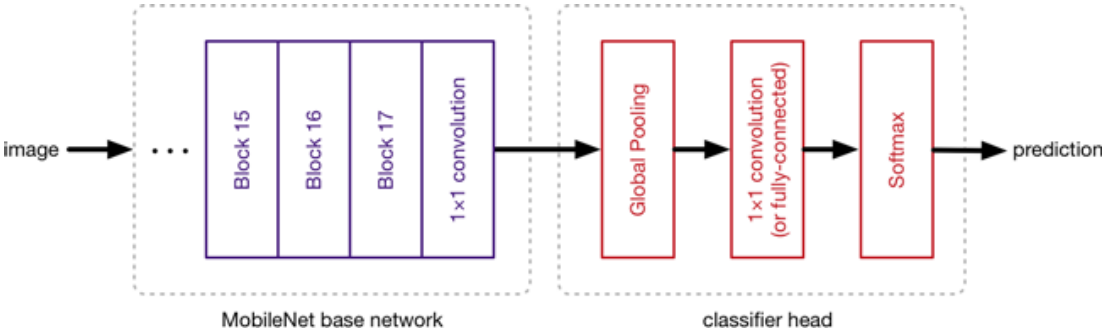



Figure 2.5: MobileNet SSD v2 Object Detection Model

ceptional speed, achieving a processing rate of 40 frames per second (FPS). This substantial improvement over previous work (25 FPS) highlights its suitability for real-time applications demanding fast and accurate object detection. While the YOLOv4 model initially exhibited lower speed compared to MobileNet-SSD at 1.7 FPS, applying the TensorRT engine optimization technique drastically improved its performance. The optimized model achieved a processing speed of 7.2 FPS, significantly faster than the original version. Although its speed remains slightly lower than MobileNet-SSD, the YOLOv4 model possesses the advantage of detecting smaller objects, making it a valuable option for applications requiring high precision in identifying intricate objects with small sizes.

### 2.1.6 Object Detection for Night Vision using Deep Learning Algorithms

This research paper [6] focuses on applying Deep Learning algorithms to address the challenge of object detection in night-time surveillance scenarios, where low illumination significantly hinders performance. Surveillance systems play a vital role in security and monitoring, but their effectiveness often diminishes in low-light conditions. Deep Learning has emerged as a powerful tool for object detection, achieving remarkable results in daytime scenarios. However, replicating this success in night-time surveillance requires innovative approaches due to the inherent limitations of lighting. The research proposes a novel approach that leverages thermal infrared images instead of traditional visible-light images. Thermal images capture heat signatures, allowing objects to be distinguished from their surroundings regardless of

ambient lighting conditions. The chosen Deep Learning model will be trained on a dataset of thermal infrared images, learning to automatically extract relevant features for object detection and classification. The research aims to unlock the potential of Deep Learning for object detection in night-time environments. The key objective is to develop a model capable of accurately identifying and classifying objects in thermal infrared images captured during night-time surveillance. This would significantly enhance the capabilities of surveillance systems, enabling them to operate effectively even in low-light conditions.

The research focuses on two primary tasks:

- Object Detection: Identifying the presence and location of external objects within a designated isolation area.
- Object Classification: Categorizing detected objects as either human or animal.

To find the best deep learning algorithm for object detection in night vision, the study analyzed various algorithms based on performance. The research selected one-stage detectors YOLO, SSD, and RetinaNet for their high inference speed, reduced computational cost, and adaptability to night vision images.

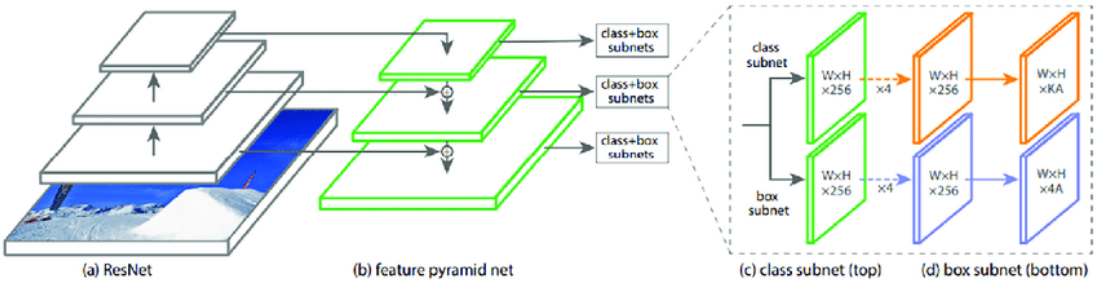



Figure 2.6: The network architecture of RetinaNet

### 2.1.7 BLIP:Bootstrapping Language- Image Pre-training for Unified Vision-Language Understanding and Generation

The paper introduces BLIP, a novel framework for Vision-Language Pre-training (VLP) that addresses limitations in existing methods by enhancing both understanding and generation tasks. BLIP utilizes noisy web data more effectively by employing

a captioner to generate synthetic captions and a filter to remove noisy ones. This process, called CapFilt, significantly improves performance across various vision-language tasks.

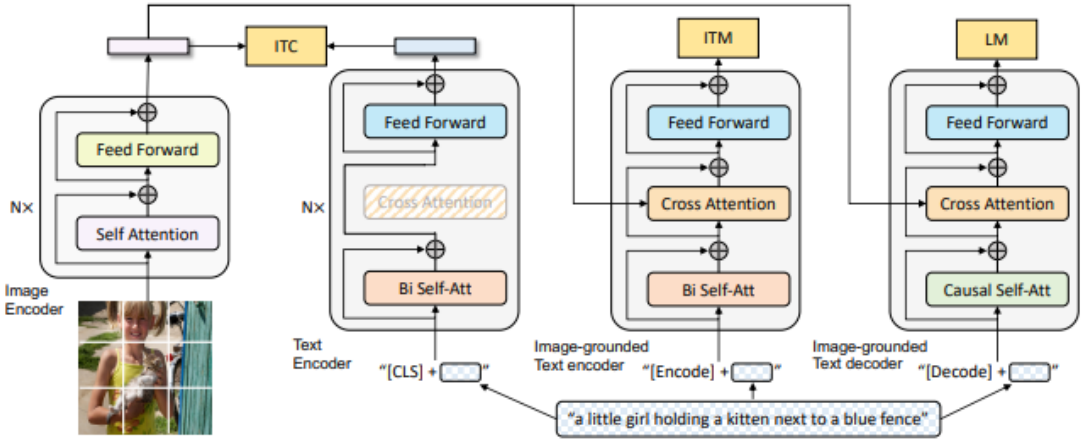



Figure 2.7: Architeture of Blip

1. **Multimodal mixture of Encoder-Decoder (MED)**: BLIP introduces a new model architecture that can function as a unimodal encoder, an image-grounded text encoder, or an image-grounded text decoder. This architecture is jointly pre-trained with three objectives: image-text contrastive learning, image-text matching, and image-conditioned language modeling.

2. **Captioning and Filtering (CapFilt)**: BLIP utilizes CapFilt for dataset bootstrapping, where a captioner generates synthetic captions and a filter removes noisy ones, resulting in a higher-quality text corpus.

Experimental results demonstrate that BLIP achieves state-of-the-art performance on various vision-language tasks, including image-text retrieval, image captioning, and visual question answering. Furthermore, BLIP exhibits strong generalization ability when directly transferred to video-language tasks in a zero-shot manner.The paper also discusses the effectiveness of nucleus sampling over beam search for generating synthetic captions, as well as the impact of parameter sharing and decoupling between the captioner and filter during pre-training.Overall, BLIP represents a significant advancement in VLP, offering improved performance and flexibility in handling both understanding and generation tasks.

### 2.1.8 Experiment on Producing Disparity Maps From Aerial Stereo Images Using Unsupervised and Supervised Methods

Disparity maps, fundamental in stereo vision, have garnered increasing attention, especially with advancements in artificial intelligence (AI) and deep learning techniques. Despite decades of research, recent statistics from Web of Science (2022) indicate a growing trend in the exploration of this domain, particularly within AI. The surge in interest is attributed to AI's capacity to learn, refine, and execute tasks with precision, particularly through deep learning methodologies.

Traditionally, generating disparity maps involved labor-intensive manual processes. However, the advent of deep learning has revolutionized this landscape, enabling machines to autonomously learn and improve their performance in tasks such as stereo image processing. One of the key advantages of employing deep learning techniques is its inherent intelligence, allowing machines to acquire knowledge and skills without explicit human intervention, thus streamlining solution acquisition.

Recent years have witnessed a proliferation of research endeavors focused on leveraging deep learning for stereo image tasks. This surge underscores the need to evaluate the efficacy of traditional and contemporary methodologies in this domain. Understanding the strengths and limitations of both approaches is crucial for advancing stereo image processing techniques effectively.

A disparity map essentially visualizes the spatial disparities between corresponding pixels in stereo image pairs. These maps are indispensable for tasks like depth estimation, 3D reconstruction, and scene understanding. By quantifying the differences between corresponding pixels in left and right images, disparity maps facilitate the extraction of depth information, enabling machines to perceive and interpret spatial relationships within a scene. The Geometry and Context Network (GC-Net) is a deep

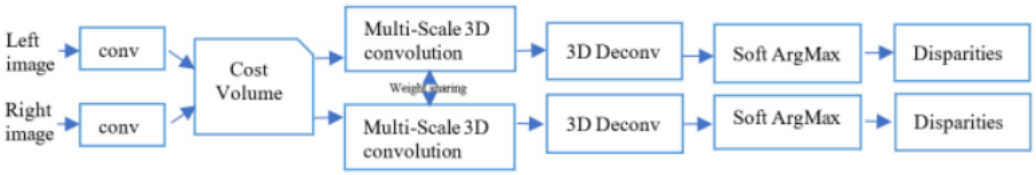



Figure 2.8: Architecture overview of GC-Net (Kendall et al., 2017).

learning architecture designed by Kendall et al. in 2017 specifically for estimating 3D geometry from stereo images. Fig. 4 provides an overview of the network architecture, which comprises several key components:

1. Convolutional Layers: The left and right stereo images are inputted into the network, undergoing a series of 2-D convolutional operations. Each convolutional layer is followed by batch normalization and rectified linear unit (ReLU) activation functions.

2. Stereo Matching Cost Computation: Following the deep representation learning stage, the network computes the stereo matching cost through additional 2-D convolutional operations.

3. Subsampling and Residual Blocks: A 5x5 convolutional filter with a stride of two is applied to subsample the input. Subsequently, eight residual blocks are appended to the network architecture. These residual blocks consist of two consecutive 3x3 convolutional filters.

4. Shared Parameters: Parameters between the left and right towers of the GC-Net are shared to facilitate the effective learning of corresponding features. This sharing of parameters enhances the network's ability to understand the relationships between the left and right stereo images.

The highlighted features of the GC-Net architecture include:

- Incorporating Context: The network directly integrates context from the input data, enabling it to capture broader spatial relationships and dependencies within the stereo images.

- 3-D Convolutions: GC-Net employs 3-D convolutions to filter the cost volume, allowing it to capture both spatial and cross-dimensional information effectively.

- Soft Argmin Function: The network utilizes a differentiable soft argmin function to regress sub-pixel disparity, enabling it to predict fine-grained depth information with greater precision.

The GC-Net architecture is implemented using the PyTorch framework and trained using a supervised regression loss function. This loss function minimizes the absolute error between the ground truth disparity and the model's predicted disparity for each pixel in the stereo image pair. By training the model on a large dataset with ground truth disparities, GC-Net learns to accurately estimate 3D geometry from stereo image pairs, making it a powerful tool for tasks such as depth estimation and 3D reconstruction.

In summary, while traditional methods for generating disparity maps have laid the foundation for stereo vision research, the advent of deep learning has sparked a new wave of innovation in this field. The integration of AI techniques promises to streamline and enhance stereo image processing tasks, opening avenues for more efficient and accurate scene analysis and interpretation.

## 2.2 Problem Statement

Visually impaired individuals encounter obstacles in accessing real-time visual information and navigating digital interfaces. Dependency on others for reading assistance limits their independence and educational opportunities.

## 2.3 Proposed Solutions

Our smart glasses employ an innovative Multi-Modal Perception system, seamlessly integrating advanced technologies like object detection, text recognition, and distance estimation. This sophisticated approach allows our system to gather a diverse range of information from the user's surroundings, providing a comprehensive and detailed understanding of the environment. By combining these modalities, we aim to enhance users' spatial awareness, offering a richer and more immersive experience.

At the core of our design philosophy is User-Centric Interaction. To ensure an intuitive and hands-free experience, we have implemented a voice command interface. This design choice empowers users to effortlessly control and access information using natural and familiar voice commands. This is particularly crucial for individuals with visual impairments, as traditional interfaces may present challenges. By prioritizing

user-friendly interaction, we aim to provide a seamless and accessible experience, allowing users to navigate their surroundings with ease and independence. Our commitment to user-centricity underscores our dedication to creating technology that aligns with users' needs and preferences.

# Chapter 3

# Software Requirement Specification

## 3.1 Overall Description

### 3.1.1 Product Perspective

The AI-powered smart glass is designed to assist visually impaired individuals in understanding their surroundings.Through seamless integration of image captioning, text recognition, and distance estimation technologies, the smart glass delivers real-time information about the surroundings. The stereo camera provides additional spatial awareness, while the voice command interface and text-to-speech conversion ensure a natural and hands-free interaction for the user. This comprehensive solution aims to empower visually impaired individuals by providing them with accessible and context-aware information about their environment, thereby promoting autonomy and safety.

### 3.1.2 Product Functionality

**Object Detection:** Detects and classifies objects in the user's environment in real-time. Provides information about the nature and location of detected objects.

**Text Recognition:** Extracts text from signs, labels, and other visual elements using OCR technology.Enables the conversion of visual information into machine-readable text.

**Natural Language Generation (NLG):** Utilizes BERT for NLG to generate context-aware and user-friendly statements based on the recognized text. Converts extracted

information into natural-sounding speech for user understanding.

**Distance Estimation:** Uses a stereo camera to estimate distances to objects in the surroundings. Enhances spatial awareness and provides warnings about obstacles in the user's path.

**Voice Command Interface:** Allows users to interact with the system using voice commands. Triggers specific functionalities such as reading text, providing location information, or initiating navigation.

**Text-to-Speech Conversion:** Implements Google Text-to-Speech for converting recognized text and generated statements into spoken words.Delivers auditory feedback to the user, conveying information about the environment and system responses.

## 3.2 Assumptions and Dependencies

### 3.2.1 Assumptions

1. The smart glass will be primarily used in environments with standard lighting conditions and minimal extreme weather conditions.

2. The users will be comfortable and able to effectively use the voice command interface for interaction.

3. The objects in the environment have distinct features necessary for successful detection using BLIP.

4. The text in the environment is sufficiently clear and legible for accurate recognition by Tesseract OCR.

5. The voice command interface can accurately understand and interpret user commands in various situations.

6. Assumes that the underlying technologies, including BLIP, Tesseract OCR, Stereo camera, and Google TTS, operate stably and reliably.

### 3.2.2 Dependencies

1. Depends on the compatibility of the smart glass hardware with the chosen technologies and their specific requirements.
2. Relies on the availability and compatibility of software libraries for BLIP, Tesseract OCR, and Google TTS.
3. Depends on the quality and clarity of the training data used for image captioning (BLIP), text recognition (Tesseract OCR), and distance estimation.
4. Depends on an internet connection for certain functionalities, such as accessing pre-trained models or utilizing cloud-based services for enhanced processing.

## 3.3 External Interface Requirements

### 3.3.1 Hardware Interfaces

1. Smart glasses with camera, microphone and speakers.
2. It has an audio output device to delivers voice feedback, audio descriptions of the environment, and notifications.
3. Onboard processor and memory.
4. Optional internet connectivity for cloud-based services.

### 3.3.2 Software Interfaces

1. Operating system for the smart glasses platform.
2. AI models for object detection, text recognition, and other functionalities.
3. Algorithms for obstacle detection,and user interaction.
4. Speech synthesis engine.
5. User interface and interaction system.

# Chapter 4

# Methodology

## 4.1 Image Captioning with BLIP

BLIP, is Bootstrapping Language-Image Pre-training, which is an AI model that excels at understanding the relationship between images and language. BLIP analyzes images and generates captions describing their content. This makes it ideal for applications like your smart glasses for visually impaired users. It combines computer vision and natural language processing techniques. It can extract visual features from images and translate them into human-readable language. It can handle noisy web data by using a bootstrapping technique. This ensures the captions it generates are accurate and reliable. They can achieve impressive results on image captioning tasks compared to other models.Overall, BLIP's ability to bridge the gap between image analysis and language generation makes it a valuable tool for tasks that require understanding visual content.Blip architecture includes the following:

- **Multimodal Mixture of Encoder-Decoder (MED)**:This is the core of BLIP's architecture. It allows the model to function in three different modes:

  Unimodal Encoder: Analyzes images only, extracting visual features.

  Image-Grounded Text Encoder: Takes an image and text caption as input, allowing the model to learn the relationship between visual content and language.

  Image-Grounded Text Decoder: Takes encoded image features and generates a textual caption describing the image.

- **Vision Encoder:**This component, often utilizing a pre-trained Vision Transformer (ViT), processes the input image. ViT breaks the image into smaller patches and encodes each patch into a vector representation.These embeddings capture the visual information within each patch.Transformer layers are applied to these embeddings, enabling the model to learn relationships between different parts of the image.

- **Text Encoder**:BLIP can incorporate a pre-trained Transformer-based language model (like BERT) for this stage. This encoder processes the textual caption and extracts its meaning representation.

- **Fusion Layer**:If both a text encoder and image encoder are used, a fusion layer combines the encoded information from both modalities (image and text).

- **Text Decoder**:This component leverages a pre-trained Transformer architecture.It takes the encoded image feature as input.The decoder utilizes its knowledge of language to generate a sequence of words, forming the image caption.

BLIP excels at image captioning, making it a valuable tool for your AI-powered smart glasses for visually impaired users. The different steps include:

### 4.1.1 Image Capture and Preprocessing:

The smart glasses utilize the built-in camera to capture live video or individual images based on user needs or pre-programmed triggers.Then the captured image undergoes preprocessing to ensure compatibility with BLIP's input format. This might involves adjusting the image dimensions to match BLIP's expected size (e.g., resizing to a square format), balancing color channels for optimal image analysis and removing artifacts or distortions that could hinder accurate caption generation.

### 4.1.2 BLIP Model Architecture:

BLIP utilizes a multimodal mixture of encoder-decoder architecture:

- Encoder (Vision Model):Analyzes the preprocessed image, extracting visual features like shapes, objects, colors, and their spatial relationships.BLIP often

uses pre-trained vision models like Vision Transformer (ViT) for this task.

- Decoder (Language Model):Takes the extracted visual features from the encoder as input.Utilizes its knowledge of language to translate these features into a sequence of words, forming the image caption.BLIP might leverage pre-trained language models like Transformer-based architectures for this step.

- Attention Mechanism:BLIP employs an attention mechanism within the encoder.This allows the model to focus on specific parts of the image most relevant to generating the caption.For example, if the image contains a person holding a cup, the attention mechanism will focus on the person's hand and the cup to generate a description like "a person holding a cup."

### 4.1.3 Inference and Caption Generation:

The preprocessed image is fed into the BLIP model running either on the smart glasses themselves or on a connected device with more processing power (like a smartphone). BLIP then performs inference, meaning it analyzes the image using its trained parameters.Based on the extracted visual features and its language understanding, BLIP generates a textual caption describing the scene in the image.

### 4.1.4 Text-to-Speech Conversion :

The generated text caption can be converted into natural-sounding speech using a Text-to-Speech (TTS) engine. This allows the user to hear the image description through the smart glasses' speakers or headphones.The TTS engine selection should prioritize clarity and natural-sounding voice for optimal user experience.

By continuously capturing and processing images, BLIP can provide real-time assistance to the user.Users can hear descriptions of their surroundings, including objects they encounter, people they interact with, or even text on signs or documents. This information can be delivered through voice prompts

## 4.2 Text Recognition using Tesseract OCR

Tesseract OCR (Optical Character Recognition) is an open-source tool developed by Google for recognizing text in images. It is designed to convert images containing printed or handwritten text into machine-readable text data. Tesseract OCR has gained popularity due to its accuracy and availability as a free and open-source software.It supports a wide range of languages, making it a versatile tool for text recognition in various linguistic contexts.Tesseract OCR can be trained for specific fonts or languages, allowing users to adapt it to their particular requirements.Tesseract OCR has shown competitive accuracy in recognizing text from images, especially for printed text. However, its performance may vary depending on the quality of the input images and the complexity of the text layout. It is widely used in applications where text needs to be extracted from images or scanned documents.

### 4.2.1 Image Capture

Using the camera on the smart glass the images of the surroundings is captured.

### 4.2.2 Preprocessing

Preprocessing this captured images to enhance the quality of the text for better recognition. This may involve tasks like resizing, cropping, or adjusting the contrast.

### 4.2.3 Text Detection

Implement a text detection algorithm to identify regions of interest (ROIs) within the images where text is present. This step helps narrow down the focus for text recognition.

### 4.2.4 Tesseract OCR Integration

Utilize the Tesseract OCR engine to recognize text within the identified ROIs. This involves passing the preprocessed images to the OCR engine and obtaining the recognized text as output.

### 4.2.5 Text-to-Speech Conversion

Once the text is recognized, convert it into speech using text-to-speech (TTS) engine. This enables the smart glass to audibly relay the information to the user.

### 4.2.6 User Interface

A user-friendly interface is designed for the smart glass, providing feedback to the user about the recognized text. This could involve spoken instructions, vibration patterns, or other sensory feedback.

## 4.3 StereoBM for distance estimation

Stereo vision relies on two cameras to simulate the depth perception of human vision. By comparing the disparities (shifts) between corresponding points in the two images, you can estimate the distance to objects in the scene.

Block Matching (BM) is a technique used within stereo vision algorithms. It involves dividing the images into blocks and comparing these blocks between the left and right images to find matching areas. The disparity of the matching blocks gives information about the depth.

The Stereo Block Matching (SBM) algorithm works by comparing blocks of pixels between the left and right images and finding the disparity using various methods like Sum of Absolute Differences (SAD) or Sum of Squared Differences (SSD). This disparity can then be used to calculate depth information.
Implementation of stereoBM

1. Camera Calibration: This step is crucial for accurate depth estimation. Camera calibration involves determining the intrinsic parameters of each camera (such as focal length, principal point, and lens distortion) and the extrinsic parameters (the relative pose of the cameras). Calibration is typically done using a calibration pattern with known geometry, like a checkerboard. Once calibrated, you can undistort the images, ensuring that straight lines remain straight and the scale is consistent across the image.

2. Rectification: In stereo vision, rectification is performed to transform the stereo image pair so that corresponding points lie on the same scanline, simplifying the stereo matching process. This transformation ensures that epipolar lines (lines along which corresponding points must lie) are parallel to the horizontal axis. Rectification can be done using the intrinsic and extrinsic parameters obtained during camera calibration.

3. Block Matching Algorithm: Block matching is a fundamental technique in stereo vision. The basic idea is to divide the rectified images into small blocks and search for corresponding blocks between the left and right images. This search can be performed using different metrics, such as Sum of Absolute Differences (SAD), Sum of Squared Differences (SSD), or normalized cross-correlation. The block with the lowest disparity indicates the best match.

4. Depth Map Calculation: Once the correspondences (disparities) between the blocks are found, you can calculate the depth map. The depth at each pixel is inversely proportional to its disparity. A simple formula to compute depth from disparity is $Z = \frac{f \times T}{d}$, where $Z$ is depth, $f$ is the focal length of the camera, $T$ is the baseline distance between the cameras, and $d$ is the disparity.

5. Finement: Depending on the accuracy required, you might need to refine the depth map further. Techniques like Semi-Global Matching (SGM), which consider the global context of disparities, can be applied to improve the quality of the depth map. Additionally, post-processing methods such as hole-filling or bilateral filtering can be used to smooth the depth map and remove noise.

## 4.4 Voice command interface

A voice command interface is a technology that enables users to interact with devices or systems using spoken commands. Instead of using traditional input methods like typing on a keyboard or tapping on a touchscreen, users can control and command devices through their voice. This interface leverages speech recognition technology to convert spoken words or phrases into actionable commands.Voice commands provide

a convenient and accessible way for people with physical limitations or disabilities to interact with technology.

### 4.4.1 Speech Recognition

The smart glasses utilize a connected microphone to capture the user's voice commands. These commands are then sent to a speech recognition engine, which converts the spoken words into digital text.

### 4.4.2 Natural Language Processing

Once the speech is recognized, a natural language processing (NLP) module analyzes the text to understand the user's intent and desired action. This involves identifying keywords, interpreting sentence structure, and analyzing context to determine the meaning of the command.

### 4.4.3 Functionality Execution

Based on the interpreted intent, the voice command interface triggers the appropriate action within the smart glasses. This could include:

1. Image captioning: Activating BLIP to identify objects in the user's field of view and generate the corresponding caption and provide audio feedback.

2. Text recognition: Triggering Tesseract OCR to extract text from captured images and convert it to audio.

3. Information retrieval: Answering user questions about their surroundings or providing specific information based on their location.

### 4.4.4 Voice Feedback

After executing the user's command, the smart glasses provide audio feedback through built-in speakers or headphones. This feedback can includes the , information requested by the user, descriptions of objects and surroundings.

# Chapter 5

# Design

## 5.1 System Architeture Design

The user interaction with the smart glasses is designed to be intuitive and responsive, centered around voice commands. It begins with the user initiating a voice command, setting the stage for seamless communication between the individual and the smart glasses.

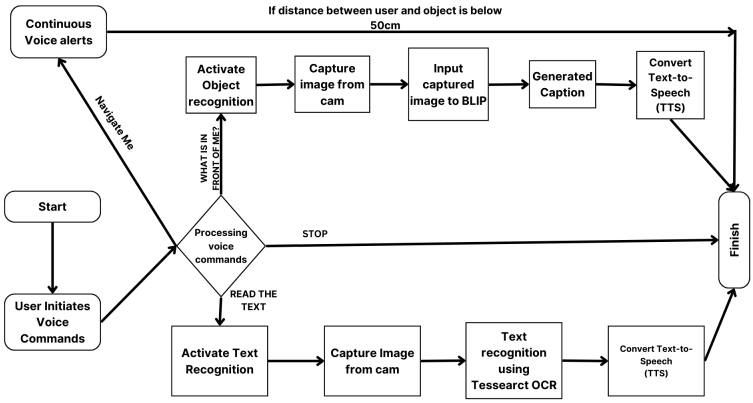



Figure 5.1: System Design

**Processing Voice Commands:** Upon receiving a voice command, the system processes the spoken input, identifying the user's intent through natural language processing (NLP) algorithms. This step ensures accurate and effective interpretation of user instructions.

**Voice Commands:** There are four primary voice commands designed for user

convenience:

**"What's in front of me?" (Object Recognition):**

- Upon this command, the object recognition module is activated.
- BLIP, optimized for edge devices, identifies objects in the user's surroundings.
- The recognized objects and their corresponding distances are then sent to BERT for natural language generation.
- The generated natural language is converted into speech using Google TTS, providing the user with real-time, context-aware information.

**"Read the text." (Text Recognition):**

- This command triggers the text recognition module, powered by Tesseract OCR.
- Text from various sources, such as documents or signs, is recognized.
- The identified text is converted into speech using Google TTS, enabling the user to access printed information effortlessly.

**"Stop" (Interrupt Command):**

- The user has the ability to halt ongoing processes by using the "STOP" voice command.
- This ensures that the system is responsive to user needs, allowing for immediate interruption or transition.

**"Navigate Me":**

The project incorporates a stereo camera that continuously monitors the distance between the user and nearby objects. When the distance falls below a predefined threshold (e.g., 50cm), the system issues a voice alert with a continuous beep, serving as a warning without the need for a specific command.

# Chapter 6

# Technology Stack

## Hardware Components

## 6.1 NVIDIA Jetson Nano

The NVIDIA Jetson Nano is a small, affordable, and powerful single-board computer designed for AI and edge computing applications. It packs the processing power of modern AI into a compact and affordable package, making it ideal for a wide range of projects and applications.The NVIDIA Jetson Nano empowers our AI-powered smart glasses for the visually impaired by running complex AI models like BLIP and Tesseract OCR, enabling image captioning and text extraction for information access. Providing powerful processing capabilities within a compact form factor, enabling integration into the smart glasses without hindering weight or battery life. It offers an open-source platform with a large and active community, facilitating project development and access to valuable resources.


Figure 6.1: NVIDIA Jetson Nano

## 6.2 Waveshare IMX219-83 Stereo Camera

he Waveshare IMX219-83 is a powerful stereo camera module designed for applications that require depth perception and three-dimensional (3D) understanding of a scene.This camera boasts two high-quality Sony IMX219 image sensors, each capturing 8 megapixel images. These sensors are positioned slightly apart, mimicking human eyes, to create a stereo vision setup. By capturing the scene from two viewpoints, the IMX219-83 enables the calculation of disparity, the difference in corresponding points between the left and right images. Disparity is inversely proportional to the distance of an object from the camera. This information is crucial for reconstructing a 3D representation of the environment and estimating object distances. The IMX219-83 is well-suited for various AI applications that leverage depth perception and stereo vision. These includes facial recognition, object recognition and tracking ,road marking and lane recognition etc.The IMX219-83 is compatible with various developer boards, including NVIDIA Jetson Nano series , Raspberry Pi etc .

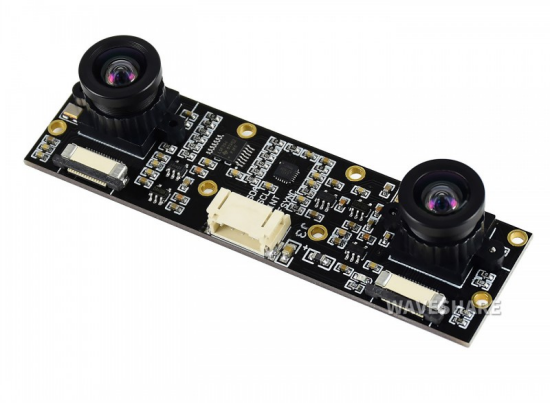


Figure 6.2: Waveshare IMX219-83 stereo Camera

## 6.3 Earphone and Microphone

Earphones are small audio devices designed to be worn inside the ears. They consist of tiny speakers (drivers) that deliver audio directly into the ears. Earphones are a popular

choice for personal audio listening due to their compact size, portability, and ability to provide a degree of noise isolation.A microphone is a device that converts sound waves into electrical signals. It captures audio and converts it into an electrical signal that can be transmitted, recorded, or amplified . The primary function of earphones in our smart glasses is to provide audio feedback to the user. As the AI model identify objects in the environment, their descriptions are converted to speech and delivered through the earphones. When text is captured by the camera and processed using Tesseract OCR, it is converted to audio and presented to the user through the earphones.The microphone allows users to interact with the smart glasses through voice commands. Control various functionalities of the smart glasses, such as activating object detection, text recognition etc.

Figure 6.3: Earphone and Microphone

# Software Components

## 6.4 Google Text-To-Speech

Google Text-to-Speech (TTS), also known as Cloud Text-to-Speech, is a Google Cloud platform service that converts text into natural-sounding speech. It uses advanced machine learning algorithms to analyze and synthesize speech, offering a wide range of features and benefits for various applications.Supports over 200 languages and dialects, enabling global reach and accessibility. Our smart glasses capture information through various sensors and the processed text is sent to the Google TTS API, which utilizes machine learning models to analyze the text and generate corresponding speech audio.

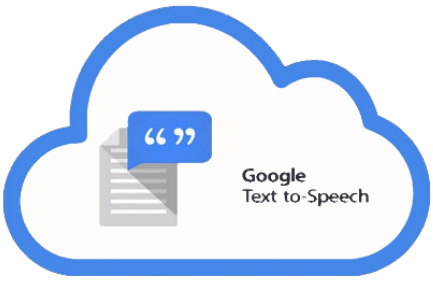



Figure 6.4: Google TTS

## 6.5 Python

Python is a high-level, general-purpose programming language known for its simple and elegant syntax, making it easy to learn and use. Unlike compiled languages, Python code is executed directly without needing compilation, making development faster and more iterative.Python scripts automate various tasks in our smart glasses system, such as data processing and analysis. Preprocessing image data captured by the camera, extracting relevant information, and preparing it for AI models.Interfacing with sensors, actuators, and other hardware components to control the smart glasses' functionality. Responding to user voice commands, generating audio feedback, and managing the overall user experience. Python provides libraries and frameworks that facilitate the integration of AI models in our smart glasses.


Figure 6.5: Python

## 6.6 Jupyter

Jupyter is an open-source web application that allows users to create and share documents containing live code, equations, visualizations, and narrative text. It is a popular tool for interactive computing, data science, and scientific computing.Users can write and execute code, visualize data, and see the results immediately, facilitating exploration and experimentation.Users can write and execute code, visualize data, and see the results immediately, facilitating exploration and experimentation.


Figure 6.6: Jupyter Notebook

## 6.7 Open CV

OpenCV (Open Source Computer Vision Library) is a powerful and widely used open-source library for computer vision, image processing, and machine learning. It provides a comprehensive set of algorithms and tools for various tasks. In our smart glass OpenCV libraries capture images and video frames from the smart glasses' camera. OpenCV filters out noise and improves image quality for better analysis by AI models.


Figure 6.7: Open CV

# Chapter 7

# Implementation

## 7.1 Text Recognition using Tesseract OCR

**Algorithm 1** Activate Text Recognition and Convert to Speech

```
0:  procedure activateTextRecognition
1:      Print "Activating text recognition..."
2:      Define configuration for text recognition as 'digits'
3:      Function performTextRecognition(imagePath):
4:          Function performTextRecognition(imagePath):
5:              Load image from imagePath using OpenCV
6:              Perform text recognition using Tesseract OCR
7:              Return recognized text
8:      Function textToSpeech(text, language='en'):
9:          Function textToSpeech(text, language='en'):
10:             Create gTTS object with text and specified language
11:             Save generated speech as MP3 file
12:             Play audio using default system audio player
```

**Algorithm 2** Main Execution

```
1: Call autoFocusCapture() function to capture image
2: Set capturedImagePath to "captured_image.jpg"
3: Set recognizedText to performTextRecognition(capturedImagePath)
4: Print "Recognized Text:"
5: Print recognizedText
6: Call textToSpeech(recognizedText) =0
```

1. **'perform_text_recognition(image_path)' Function**:This function aims to recognize text from an image file using Optical Character Recognition (OCR)

technology.It takes the file path of the image (image_path) as a parameter.It uses the OpenCV library (cv2) to read the image from the specified path and load it into memory as an image object (img).Then, it utilizes the pytesseract library, which is a Python wrapper for Google's Tesseract-OCR Engine. Tesseract is a widely-used OCR engine capable of recognizing text from images.The imag_to_string function from pytesseract is called with the loaded image (img) as input to perform OCR and extract text.The recognized text extracted from the image is returned by the function.

2. **'text_to_speech'(text, language='en') Function**:This function converts recognized text into speech.It takes the recognized text (text) as input along with an optional parameter language (default is 'en' for English).It utilizes the gTTS library, which stands for Google Text-to-Speech. This library provides an interface to Google's Text-to-Speech API. A gTTS object is created with the recognized text and the specified language.The save method is called to save the generated speech as an MP3 audio file named "speech.mp3".The os.system function is used to play the generated audio file using the default system audio player.The speech corresponding to the recognized text is played.

3. **Main Function**

- auto_focus_capture(): This function is called before performing text recognition.
- Perform Text Recognition:After capturing the image, the perform_text_recognition function is called with the path of the captured image (captured_image_path) as input.The recognized text extracted from the image is stored in the variable recognized_text.
- Print Recognized Text:The recognized text is printed to the console using the print function.
- Text-to-Speech Conversion:Finally, the text_to_speech function is called with the recognized text as input. This converts the recognized text into speech and plays it using the system's default audio player.

Thisdemonstrates a workflow where text is extracted from an image using OCR and then converted into speech. This can be useful for visually impaired users by providing audio descriptions of text in images or automating tasks involving the processing of textual information from images.

## 7.2 Object Recognition and Image Captioning using Blip

**Algorithm 3** Activate Object Recognition

1: **FUNCTION** activateObjectRecognition
2: Import necessary libraries: requests, PIL.Image, transformers.BlipProcessor, transformers.BlipForConditionalGeneration
3: Load pre-trained models: processor, model ← LoadModels(”Salesforce/blip-image-captioning-large”)
4: Specify image file path: image_path ← ”captured_image.jpg”
5: Open and preprocess the image: raw_image ← OpenImage(image_path)
6: Perform unconditional image captioning: inputs ← PreprocessImage(processor, raw_image)
7: Generate image caption: caption ← GenerateCaption(model, inputs)
8: Output caption =0

1. **’activate_object_recognition()’ Function**: This function activates object recognition and generates a caption for an image using a pre-trained transformer-based model. The ’requests’ library allows the code to make HTTP requests, which is used to download the pre-trained model.’PIL’ (Python Imaging Library) is used for opening and processing images and ’transformers’ provides an interface to work with transformer-based models, including BERT, GPT, etc.

- Model Loading: The function loads the BlipProcessor and BlipForConditionalGeneration models from the ”Salesforce/blip-image-captioning-large” checkpoint using the ’from_pretrained’ method. These models are pre-trained on large datasets for image captioning tasks.

- Image Processing: The function expects the file path of the image (’image_path’) as input. You should replace it with the actual file path where the image is located.The image is opened using PIL (’Image.open(image_path).convert(’RGB’)’)

and converted to the RGB model. This conversion ensures that the image is in the appropriate format for processing.

- Unconditional Image Captioning: The processor processes the raw image using the loaded BlipProcessor model to prepare it for input to the transformer model ('processor(raw_image, return_tensors="pt")').The processed inputs are then passed to the BlipForConditionalGeneration model for caption generation ('model.generate'inputs)').The generated caption is printed to the console after decoding it using the processor ('processor.decode(out[0], skip_special_tokens=True)').

This algorithm demonstrates how to activate object recognition and generate a caption for an image using a pre-trained transformer-based model specifically designed for image captioning tasks.It illustrates the usage of transformer-based models for image captioning tasks, showcasing the capabilities of AI in understanding and describing visual content. By combining machine learning with image processing techniques, the code demonstrates how to leverage pre-trained models to generate natural language descriptions for images.

## 7.3 Distance Estimation using StereoBM

### 7.3.1 Starting the Camera

**Algorithm 4** Start the Camera

```
Initialize video capture, frame, grabbed, read thread, read lock, and running
Generate gstreamer pipeline string
Open cameras using generated pipeline
if running then
    Print ”Video capturing is already running” and return
end if
if video capture element then
    Set running as True and start a new thread to update camera images
end if
Set running as False and join read thread
if video capture element then
    Release it
end if
if read thread then
    Join it
end if
Generate gstreamer pipeline string based on provided parameters
Return pipeline string =0
```

This algorithm outlines the process of initializing and managing camera operations for image capture. It first sets up essential variables and initializes the camera with a specified gstreamer pipeline string. It checks if the camera is already running to prevent duplicate operations. If the camera is available, it starts a new thread to continuously update the captured images. After the camera usage is complete, it releases the camera resources and stops the image update thread to free up system resources. This algorithm ensures efficient and controlled camera usage for image acquisition tasks.

### 7.3.2 Taking the picture

**Algorithm 5** Take Pictures

```
Function TakePictures
Prompt user to start image capturing
if user input is 'y' then
    Start left and right cameras
    Create a window for displaying images
    Initialize counter and timer
    while counter is less than or equal to total photos do
        Update countdown timer
        Read images from left and right cameras
        if images are grabbed successfully then
            Combine images
            if countdown timer reaches 0 then
                Increment counter
                Save images
                if images folder exists then
                    Save image
                else
                    Create images folder
                    Save image
                end if
                Reset timer
                Wait for a few seconds
            end if
            Display images with countdown timer
            Wait for user input
            if user presses 'q' then
                Exit loop
            end if
        end if
    end while
else if user input is 'n' then
    Print "Quitting!"
    Exit program
else
    Print "Please try again!"
end if
Stop and release cameras
Close image window =0
```

This algorithm outlines the process of capturing images using left and right cameras. It prompts the user to start the image capturing process and initializes the necessary variables. The algorithm continuously reads images from both cameras, combines them, and displays them with a countdown timer. When the timer reaches zero, the images are saved, and the process repeats until the desired number of photos

is captured. The user can quit the program at any time by pressing 'q'. Finally, the cameras are stopped and released, and the image window is closed.

### 7.3.3 Selection of the image

**Algorithm 6** Image selection

```
Initialize photo counter
if directory 'pairs' does not exist then
    Create directory 'pairs'
end if
while photo counter is not equal to total photos do
    Initialize variable k
    Construct filename for the current photo
    if file with filename does not exist then
        Print "No file named filename"
        Increment photo counter and continue
    end if
    Read the pair image
    Display the pair image
    Wait for user input
    if user presses 'y' then
        Split the pair image into left and right images
        Save left and right images
        Increment photo counter
        Print "Pair No saved."
    else if user presses 'n' then
        Increment photo counter
        Print "Skipped"
    else if user presses 'q' then
        Break the loop
    end if
end while
Print "End cycle" =0
```

This algorithm is designed to process a set of images captured for rectification. It iterates through each image, displaying it to the user. The user is prompted to accept or skip the image based on its quality. If accepted, the image is split into left and right images and saved accordingly. If skipped, the algorithm moves on to the next image. The process continues until all images have been reviewed.

### 7.3.4 Stereo Camera Calibration and Rectification

**Algorithm 7** Calibration and Rectification

```
Set total number of photos to be used for calibration
Set chessboard parameters: rows, columns, and square size
Initialize StereoCalibrator with chessboard parameters and image size
Initialize photo counter to 0
while photo counter is less than total photos do
    Increment photo counter
    Read left and right image pairs
    if left and right images exist then
        Resize right image to match dimensions of left image
        Attempt to find chessboard corners in both images
        if corners found in both images then
            Add corners to calibrator and display them
        else
            Print error message and ignore pair
        end if
    else
        Print pair not found message and continue to next pair
    end if
end while
Complete calibration process and export results
Read calibration results
Rectify last pair of images using calibration results
Display and save rectified images =0
```

The algorithm performs stereo camera calibration and rectification using a set of image pairs. It iterates through each pair, attempting to find chessboard corners in both left and right images. If successful, it adds the corners to the calibrator for calibration. After processing all pairs, it completes the calibration process and exports the results. Finally, it rectifies the last image pair using the calibration results and displays the rectified images.

### 7.3.5 Tuning the Depth Map

**Algorithm 8** Tuning Depth Map with Real-Time Video

```
Set global depth map parameters
Define stereo depth map function
Define function to save or load depth map settings from file
Define function to activate trackbars
Create trackbars for adjusting depth map parameters
Start left and right cameras
Create window for displaying depth map
Create variables and mappings for depth map parameters
while true do
    Read frames from left and right cameras
    Convert frames to grayscale
    Rectify stereo image pair
    Update depth map parameters based on trackbar positions
    Adjust parameters for proper use in stereo depth map function
    Process stereo image pair to generate depth map
    Display depth map
    Display rectified stereo image pair
    if key pressed is 'q' then
        Exit loop
    end if
end while
Stop and release cameras
Close all windows =0
```

This algorithm tunes the depth map using real-time video from stereo cameras. It initializes global depth map parameters, defines functions for depth map processing and parameter saving/loading, creates trackbars for parameter adjustment, starts the cameras, and displays the depth map and rectified stereo images in real-time. The algorithm continuously updates the depth map based on user-adjusted parameters and exits upon user command, stopping the cameras and closing all windows.

The StereoBM parameters are:

”SADWindowSize”:69,

”minDisparity”:-100,

”numberOfDisparities”:256,

”preFilterCap”:31,

”preFilterSize”:5,

”speckleRange”:0,

”speckleWindowSize”:0

, ”textureThreshold”:0,

”uniquenessRatio”:3

### 7.3.6 Distance Alert

**Algorithm 9** Real-Time Depth Map Tuning with Stereo Cameras

Initialize Parameters:
$duration = 0.1$ seconds
$freq = 440$ Hz
Global depth map parameters:
$SWS = 5$
$PFS = 5$
$PFC = 29$
$MDS = -30$
$NOD = 160$
$TTH = 100$
$UR = 10$
$SR = 14$
$SPWS = 100$
**Load Map Settings:**
Load parameters from file
Read JSON file and assign values to global parameters
Create StereoBM object with loaded parameters
**Stereo Depth Map:**
Read frames from left and right cameras
Convert frames to grayscale
Rectify frames using calibration data
Compute depth map using StereoBM algorithm
Normalize disparity values and exclude noise
Convert depth map to color
Display depth map
Play sound for specific disparity values
**Main:**
Start left and right cameras
Load map settings from file
Create a window for displaying depth map
Loop indefinitely:
Read frames from both cameras
Stop and release cameras
Close all windows =0

The algorithm initializes parameters and loads predefined depth map settings from

a file. It computes the depth map from rectified frames captured by stereo cameras, excluding noise values. Instead of displaying the depth map, it calculates the distance from the disparity map and triggers an alert sound if the distance is less than 50 cm.

## 7.4 Voice Command System

**Algorithm 10** Listen for Commands

```
FUNCTION listenForCommands
while True do {Continuous listening loop}
  with sr.Microphone() as source:
    print "Listening for commands..."
    audio ← recognizer.listen(source)

  {Recognize speech using Google Speech Recognition}
  command ← recognizer.recognize_google(audio).lower()
  print "Command recognized:", command

  if "what's in front of me" in command then
    activateObjectRecognition() {Activate object recognition}
  else if "read the text" in command then
    activateTextRecognition() {Activate text recognition}
  else if "navigate me" in command then
    activateNavigation() {Activate navigation}
  else if "stop" in command then
    print "Stopped Listening!!"
    break {Exit the loop}
  else
    print "Unknown command. Please try again."
  end if
sr.UnknownValueError
  print "Google Speech Recognition could not understand audio."
sr.RequestError as e
  print "Could not request results from Google Speech Recognition service;
0".format(e)
end while=0
```

The code defines a function listen_for_commands() that continuously listens for voice commands and performs corresponding actions.The while True loop ensures the program keeps listening for commands indefinitely. Inside the loop, the code utilizes the sr.Microphone() function from a speech recognition library (likely SpeechRecognition) to access the user's microphone. A with statement ensures proper resource

management for the microphone.The recognizer.listen(source) line captures audio from the microphone and stores it in the audio variable. This audio data serves as the input for speech recognition.

- Speech Recognition: The recognizer.recognize_google(audio).lower() line attempts to recognize the speech within the captured audio using Google Speech Recognition. The recognized text is converted to lowercase using .lower() for case-insensitive command processing.The code then prints the recognized command for the user's reference and confirmation.

- Command Processing:

  *"what's in front of me"*: This triggers a call to the activate_object_recognition() function, likely designed to activate an object detection model using the camera.

  *"read the text"*: This triggers a call to the activate_text_recognition() function, likely designed to activate a text recognition model using the camera.

  *"Navigate me"*:This triggers a call to the activate_navigation() function,likely designed to activate a distance estimation.

  *"stop":* This command breaks out of the while True loop, effectively stopping the program from listening for further commands. A message is printed to indicate termination.

Any unrecognized command leads to a message prompting the user to try again. This ensures the system only executes intended functionalities. Error Handling. The code incorporates exception handling mechanisms to manage potential errors during speech recognition: "sr.UnknownValueError" ,this exception handles scenarios where Google Speech Recognition fails to understand the audio input. In such cases, an informative message is printed and "sr.RequestError" deals with errors related to communication with the Google Speech Recognition service. An error message with details (stored in the e variable) is printed for troubleshooting purposes. The code calls listen_for_commands() to initiate the continuous listening process. This line executes the entire functionality defined within the function.

# Chapter 8

# Testing

Testing plays a pivotal role in the development lifecycle of our AI-powered smart glasses designed for visually impaired individuals, which utilize NVIDIA as the edge AI device. It involves systematically verifying and validating different aspects of the system, such as functionality, performance, reliability, and scalability, ensuring that it meets the desired requirements and operates as intended. Various testing strategies and methodologies are employed to thoroughly assess the glasses' behavior under different conditions, providing confidence in their overall effectiveness.

## 8.1 Testing Strategies

This comprehensive plan is to assess both the functionality and performance of our AI-powered smart glasses for the visually impaired, utilizing NVIDIA as the edge AI device and incorporating Tesseract OCR, BLIP, and StereoBM. Through a strategic implementation, it systematically identify and address potential issues, thereby ensuring the reliability of the system.

### 8.1.1 Unit Testing

Testing individual components or functions of the system in isolation, ensuring they behave as expected. This could involve testing the accuracy of text recognition, object detection and distance estimation models independently.

### 8.1.2 Integeration Testing

Testing the interaction between different modules or components of the system, such as the integration between text recognition, object recognition and distance estimation with voice command system.

### 8.1.3 Functional testing

Ensuring that each function or feature of the system performs according to specifications. For instance, ensuring that the distance alert system ,image captioning and text recognition males the user feasible.

### 8.1.4 Load Testing

Evaluating the system's performance under low light and bad weather. This could involve testing how well the system performs when there are frequent changes in the environment.

## 8.2 Sample Test Cases

Test cases are a collection of representative scenarios designed to validate the functionality and performance of the system.

### 8.2.1 Tesseract OCR

In evaluating the performance of our AI-powered smart glasses for the visually impaired, we conduct testing to assess their ability to accurately detect and recognize the printed text. This involves evaluating how well the glasses can identify and segment text elements, such as lines or numbers, using Tesseract OCR for text recognition.

Table 8.1: Test Cases for Tesseract OCR

| Sl No. | Test Cases | Expected Result | Test Result |
|---|---|---|---|
| 1 | Printed Text | Recognize printed text. | Recognized Successfully |
| 2 | Traffic Sign Text | Correctly reads text on traffic signs | Detected Successfully |


```
Recognized Text:
WRONG
WAY
GO BACK
```

**CHAPTER I. Down the Rabbit-Hole**

Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, 'and what is the use of a book,' thought Alice 'without pictures or conversation?'

So she was considering in her own mind (as well as she could, for the hot day made her feel very sleepy and stupid), whether the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies, when suddenly a White Rabbit with pink eyes ran close by her.

There was nothing so VERY remarkable in that; nor did Alice think it so VERY much out of the way to hear the Rabbit say to itself, 'Oh dear! Oh dear! I shall be late!' (when she thought it over afterwards, it occurred to her that she ought to have wondered at this, but at the time it all seemed quite natural); but when the Rabbit actually TOOK A WATCH OUT OF ITS WAISTCOAT-POCKET, and looked at it, and then hurried on, Alice started to her feet, for it flashed across her mind that she had never before seen a rabbit with either a waistcoat-pocket, or a watch to take out of it, and burning with curiosity, she ran across the field after it, and fortunately was just in time to see it pop down a large rabbit-hole under the hedge.

In another moment down went Alice after it, never once considering how in the world she was to get out again.

The rabbit-hole went straight on like a tunnel for some way, and then dipped suddenly down, so suddenly that Alice had not a moment to think about stopping herself before she found herself falling down a very deep well.

```
Recognized Text:
CHAPTER I. Down the Rabbit-Hole

Alice was beginning to get very tired of sitting by her sister on the
bank, and of having nothing to do: once or twice she had peeped
into the book her sister was reading, but it had no pictures or
conversations in it, 'and what is the use of a book,' thought Alice
'without pictures or conversation?'

So she was considering in her own mind (as well as she could, for
the hot day made her feel very sleepy and stupid), whether the
pleasure of making a daisy-chain would be worth the trouble of
getting up and picking the daisies, when suddenly a White Rabbit
with pink eyes ran close by her.

There was nothing so VERY remarkable in that; nor did Alice think
it so VERY much out of the way to hear the Rabbit say to itself, 'Oh
dear! Oh dear! I shall be late!" (when she thought it over
afterwards, it occurred to her that she ought to have wondered at
this, but at the time it all seemed quite natural); but when the
Rabbit actually TOOK A WATCH OUT OF ITS WAISTCOAT-POCKET,
and looked at it, and then hurried on, Alice started to her feet, for
it flashed across her mind that she had never before seen a rabbit
with either a waistcoat-pocket, or a watch to take out of it, and
burning with curiosity, she ran across the field after it, and
fortunately was just in time to see it pop down a large rabbit-hole
under the hedge.

In another moment down went Alice after it, never once
considering how in the world she was to get out again.

The rabbit-hole went straight on like a tunnel for some way, and
then dipped suddenly down, so suddenly that Alice had not a
moment to think about stopping herself before she found herself
falling down a very deep well.
```

Figure 8.1: Examples of test cases for Tesseract OCR

### 8.2.2 BLIP

Testing the Blip model for image captioning and object detection entails evaluating its capability to accurately identify and segment objects within images. This evaluation encompasses assessing its performance in detecting objects of various interest. We rely on visual inspection of detected bounding boxes to determine their accuracy. A bounding box that accurately encompasses the entire object is deemed successful.

Table 8.2: Test Cases for BLIP

| Sl No. | Test Cases | Expected Result | Test Result |
|---|---|---|---|
| 1 | A class room | Many children sitting at desk with a teacher. | Caption generated |


Figure 8.2: Example of test case for BLIP

### 8.2.3 StereoBM

Testing the stereo Block Matching (BM) model involves evaluating its ability to accurately identify and segment objects within stereo images. We rely on visual inspection of the matched disparity maps to determine the accuracy of object segmentation. A disparity map that accurately represents the depth information of objects is considered successful.

Table 8.3: Test Cases for StereoBM

| Sl No. | Test Cases | Expected Result | Test Result |
|---|---|---|---|
| 1 | Image | Depth Map and alert | Alert produced successfully |

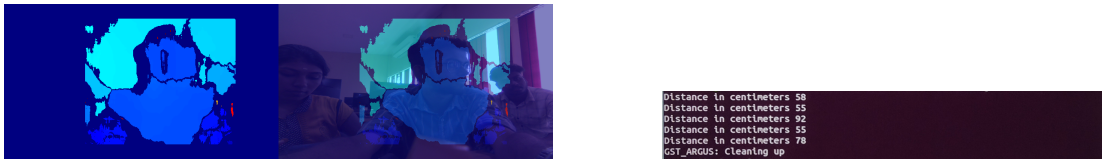



Figure 8.3: Example of test case for StereoBM

# Chapter 9

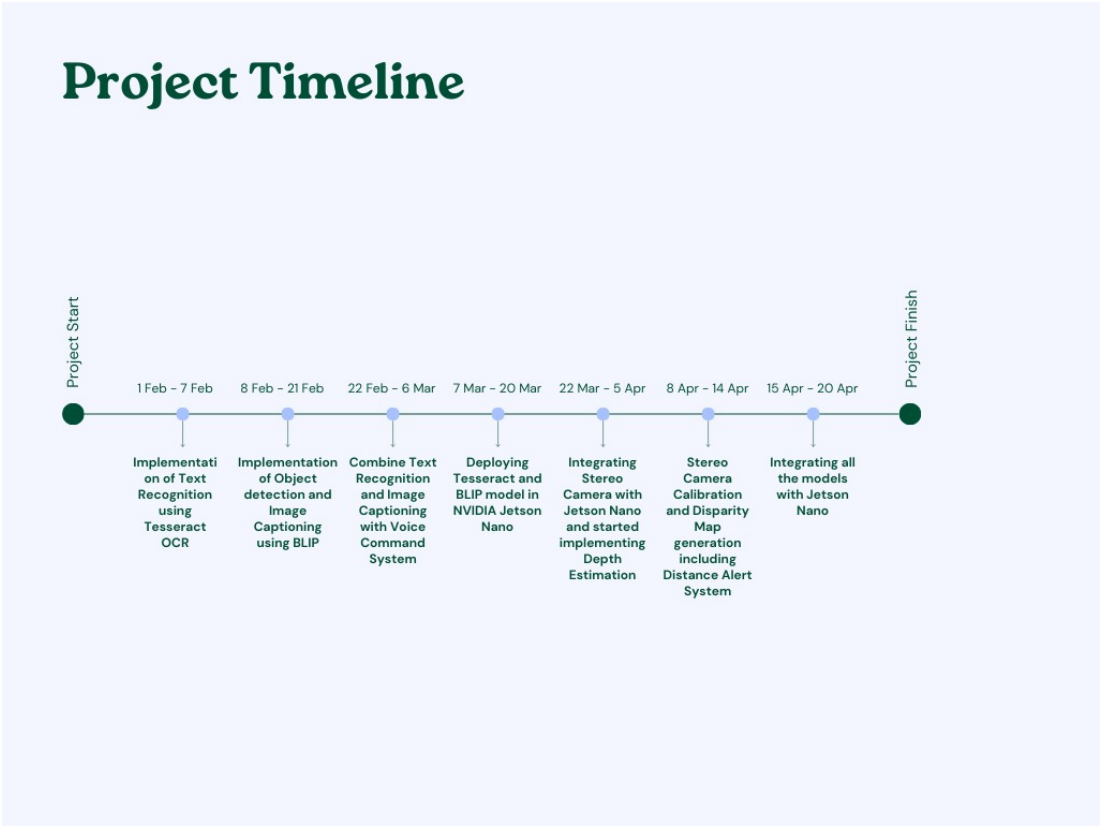
# Results and Discussions

The project successfully deployed an AI-powered smart glass system for visually impaired individuals, integrating various functionalities such as text recognition using Tesseract OCR, object detection and image captioning using Blip, and distance estimation using stereoBM. This system leverages the capabilities of the NVIDIA Jetson Nano computing device and utilizes a stereo camera for image capturing, along with earphones and a microphone for voice command input and output feed. The smart glass system accurately identifies and interprets various textual information through the Tesseract OCR function, enabling users to access essential textual content. Additionally, with the object detection and image captioning functionalities powered by Blip, the system can detect and describe objects in the user's surroundings, providing valuable contextual information. Furthermore, the integration of the stereoBM algorithm facilitates distance estimation, allowing users to perceive the spatial layout of their environment and navigate safely. The system incorporates a voice command interface, enabling users to activate and control these functionalities seamlessly through spoken commands.

However, one limitation of the system is that the voice command system can only control one aspect of the system's behavior at a time. Despite this constraint, the system demonstrates improved memory efficiency and stability, ensuring reliable functionality in dynamic environments.Moving forward, addressing this limitation through advanced reinforcement learning techniques or hybrid control strategies could further enhance the system's performance and responsiveness, ultimately advancing

the capabilities of assistive technology for visually impaired individuals in real-world scenarios.

# Project Timeline



# Chapter 10

# Conclusion

In the pursuit of enhancing accessibility and independence for individuals with visual impairments, the system has emerged as a transformative and innovative solution. Leveraging cutting-edge technologies, including BLIP for image captioning, Tesseract OCR for text recognition, and StereoBM for distance estimation , the smart glasses provide real-time, context-aware information through intuitive voice commands.

The integration of a sensor adds an additional layer of safety, alerting users to obstacles in their immediate vicinity. This comprehensive system aims not only to assist with object and text recognition but also to create a seamless and user-friendly experience through voice interactions.

By combining the strengths of various technologies, the smart glasses empower visually impaired individuals to navigate their surroundings independently, access printed and digital information effortlessly, and receive timely alerts about potential obstacles. The system represents a significant step forward in the intersection of artificial intelligence and assistive technology, embodying a commitment to improving the quality of life for those with visual impairments.

# References


[1] J. Nasreen, W. Arif, A. A. Shaikh, Y. Muhammad and M. Abdullah, ”Object Detection and Narrator for Visually Impaired People,” 2019 IEEE 6th International Conference on Engineering Technologies and Applied Sciences (ICETAS), Kuala Lumpur, Malaysia, 2019.

[2] Mahdi Safaa, A., H. Muhsin Asaad, and I. Al-Mosawi Ali. ”Using ultrasonic sensor for blind and deaf persons combines voice alert and vibration properties.” Research Journal of Recent Sciences.

[3] Topal, M. Onat et al. “Exploring Transformers in Natural Language Generation: GPT, BERT, and XLNet.” ArXiv abs/2102.08036 (2021): n. pag.

[4] Patel, Chirag Patel, Atul Patel, Dharmendra. (2012). Optical Character Recognition by Open source OCR Tool Tesseract: A Case Study. International Journal of Computer Applications. 55. 50-56. 10.5120/8794-2784.

[5] H. T. Minh, L. Mai and T. V. Minh, ”Performance Evaluation of Deep Learning Models on Embedded Platform for Edge AI-Based Real time Traffic Tracking and Detecting Applications,” 2021 15th International Conference on Advanced Computing and Applications (ACOMP), Ho Chi Minh City, Vietnam, 2021, pp. 128-135, doi: 10.1109/ACOMP53746.2021.00024.

[6] Bhabad, Dipali, Surabhi Kadam, Tejal Malode, Girija Shinde, and Dipak Bage. ”Object Detection for Night Vision using Deep Learning Algorithms”,2023 International Journal of Computer Trends and Technology

[7] G. Chandan, A. Jain, H. Jain and Mohana, ”Real Time Object Detection and Tracking Using Deep Learning and OpenCV,” 2018 International Conference on Inventive Research in Computing Applications (ICIRCA).

[8] Sudharani, B., Lokanath, K., Bhavya, G., Priya, K. H., Venkat, K., Chandu, G. R. S. ”SMART GLASSES FOR VISUALLY CHALLENGED PEOPLE USING FACIAL RECOGNITION”

[9] O. Stephen, D. Mishra and M. Sain, ”Real Time object detection and multilingual speech synthesis,” 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kanpur, India, 2019

[10] Li, J., Li, D., Xiong, C. and Hoi, S., 2022, June. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In International conference on machine learning.

[11] Zhu, L., Hattula, E., Raninen, J., and Hyyppä, J.: Experiment on Producing Disparity Maps From Aerial Stereo Images Using Unsupervised and Supervised Methods, Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci., XLVIII-4/W1-2022, 561–568, https://doi.org/10.5194/isprs-archives-XLVIII-4-W1-2022-561-2022, 2022

PAPER NAME

**AI powered smart glass for visually impai red.pdf**

| | |
|---|---|
| WORD COUNT | CHARACTER COUNT |
| **10753 Words** | **63282 Characters** |
| PAGE COUNT | FILE SIZE |
| **53 Pages** | **2.0MB** |
| SUBMISSION DATE | REPORT DATE |
| **Apr 23, 2024 12:03 PM GMT+5:30** | **Apr 23, 2024 12:04 PM GMT+5:30** |

## 9% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 7% Internet database
- 6% Publications database
- Crossref database
- Crossref Posted Content database