

GNG1103

Design Project User and Product Manual

GlassTek User Manual

Submitted by:

GlassTek Group 23

Ethan Chen, 300366920

Skylar Samm, 300445348

Kevin Mo, 300426690

Hannah Chowdhury,

Ziad Tariq,

Mulham Hajjat,

03-12-2024

University of Ottawa

Table of Contents

Table of Contents	ii
List of Figures	iv
List of Tables	v
List of Acronyms and Glossary	vi
1 Introduction.....	Error! Bookmark not defined.
2 Overview.....	1
2.1 Conventions.....	4
2.2 Cautions & Warnings.....	4
3 Getting started.....	5
3.1 Set-up Considerations	5
3.2 User Access Considerations	5
3.3 Accessing the System.....	5
3.4 System Organization & Navigation	5
3.5 Exiting the System	5
4 Using the System	6
4.1 Object Recognition.....	6
4.1.1 N/A.....	6
5 Troubleshooting & Support	7
5.1 Error Messages or Behaviors	7
5.2 Special Considerations	7
5.3 Maintenance	7
5.4 Support	7
6 Product Documentation	8
6.1 <Subsystem 1 of prototype>	8
6.1.1 BOM (Bill of Materials)	8
6.1.2 Equipment list	8
6.1.3 Instructions.....	8

6.2	Testing & Validation.....	9
7	Conclusions and Recommendations for Future Work.....	11
8	Bibliography	12
	APPENDICES	15
	APPENDIX I: Design Files	15
	APPENDIX II: Other Appendices	16

List of Figures

Figure 1. Screenshot of imports.....	2
Figure 2. Screenshot of using the OpenCV library to recognize objects.....	3
Figure 3. Screenshot of application recognizing the object.....	3
Figure 4. Screenshot of software misidentifying object.....	10
Figure 5. Same screenshot as Figure 3.....	11

List of Tables

Table 1. Acronyms.....	vi
Table 2. Glossary	vi
Table 3. Referenced Documents	14

List of Acronyms and Glossary

Table 1. Acronyms

Acronym	Definition
IDE	Integrated development environment
N/A	Not applicable
PC	Personal computer
UPM	User project manual

Table 2. Glossary

Term	Acronym	Definition

1.1.1 Introduction

This User and Product Manual (UPM) provides the information necessary for developers to effectively use GlassTek's smart glass assistant prototype and for prototype documentation. The purpose of this document is to provide adequate information for developers to pick up this project and understand how it works. This program requires access to your webcam.

1.1.2 Overview

Roughly 75% of adults in the world need corrective glasses (vision magazine), creating a need for glasses that can help users navigate the world. Shabodi tasked us with creating smart glass assistance software for this purpose.

In this project, the user and the buyer are different:

- Shabodi sells our product to companies that make smart glasses (e.g., Ray-Ban, Google).
- The users of the software are the people who purchase these smart glasses.

User Needs:

- Ease of use
- Affordability
- Comfort and style
- Strong connection

Our solution leverages free resources found online. No additional purchases are required. If Shabodi's APIs cannot be used, free bandwidth APIs can be sourced online.

Key Tools:

- OpenCV for object recognition.
- Shabodi's bandwidth API

```
import numpy as np
import tensorflow as tf

# Load a pre-trained MobileNet model from TensorFlow Hub
model = tf.keras.applications.MobileNetV2(weights='imagenet')

# Label mapping for ImageNet classes
# with open('imagenet_labels.txt', 'r') as f:
#     labels = f.read().splitlines()

# Initialize the camera
cap = cv2.VideoCapture(0)

def preprocess_frame(frame):
    # Resize the frame to (224, 224), which MobileNetV2 expects
    frame_resized = cv2.resize(frame, (224, 224))
    # Normalize and add batch dimension
    frame_preprocessed = tf.keras.applications.mobilenet_v2.preprocess_input(frame_resized)
    return np.expand_dims(frame_preprocessed, axis=0)

def recognize_objects(frame):
    # Preprocess frame and predict
    processed_frame = preprocess_frame(frame)
    predictions = model.predict(processed_frame)
    top_prediction = tf.keras.applications.mobilenet_v2.decode_predictions(predictions, top=1)[0][0]
    return top_prediction[1], top_prediction[2] # Label and confidence
```

Figure 1. Screenshot of imports.


```

while cap.isOpened():
    # Capture frame-by-frame
    ret, frame = cap.read()
    if not ret:
        break

    # Perform object recognition
    label, confidence = recognize_objects(frame)

    # Display the label and confidence on the frame
    cv2.putText(frame, f'{label}: {confidence*100:.2f}%', (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)

    # Display the resulting frame
    cv2.imshow('Object Recognition', frame)

    # Press 'q' to exit
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# When everything is done, release the capture and close windows
cap.release()
cv2.destroyAllWindows()

```

Figure 2. Screenshot of using the OpenCV library to recognize objects.

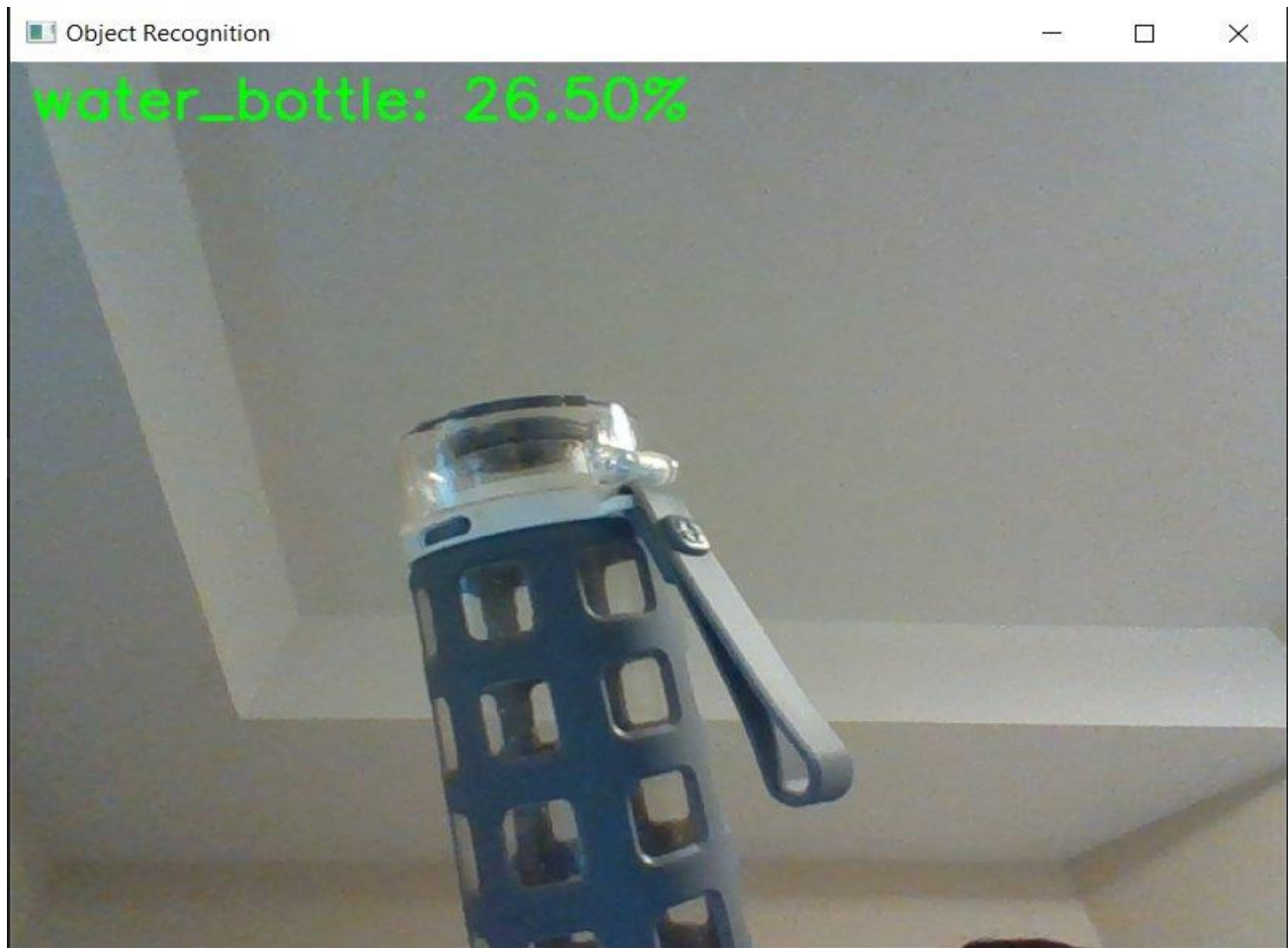


Figure 3. Screenshot of application recognizing the object.

Our solution uses OpenCV and Shabodi's bandwidth API to recognize basics objects. The code accesses the camera of the laptop.

1.2 Conventions

Action: to use the program click the run button.

1.3 Cautions & Warnings

This program requires access to OpenCV, Shabodi's token generation API, and Shabodi's bandwidth API. The object recognition is not always accurate.

2 Getting started

To get started, download the file from MakerRepo and use your choice of a Python IDE. To open the file, right-click on the file and select “Open with” and select your chosen IDE.

2.1 Configuration Considerations

This software requires a PC with an internet connection and camera.

2.2 User Access Considerations

This software and user manual is made for developers. Regular users are not suggested to use this software.

2.3 Accessing/setting up the System

Download the file and run it via your IDE of choice.

2.4 System Organization & Navigation

N/A

2.1 Exiting the System

Simply exit the application by pressing the “x” button. This is usually on the top-right for Windows users. Windows users can also use alt+F4 to exit the current application.

3 Using the System

There is only one intended way to use the system – that is to run it via your chosen IDE.

3.1 Object Recognition

Object recognition: This software uses OpenCV's object detection library to detect objects.

3.1.1 N/A

N/A

4 Troubleshooting & Support

This project has been discontinued by the original developers. There is no contacting them. For troubleshooting, consult the console for error codes.

4.1 Error Messages or Behaviors

If all the required software is downloaded correctly no error messages should show up. The only thing is that there is an accuracy display that will show how well the program is running.

4.2 Special Considerations

N/A

4.3 Maintenance

Check to see if APIs need to be updated.

4.4 Support

This project is discontinued by the original developers.

If the program doesn't work:

- Ensure that your webcam is properly connected and working.
- Verify that all required libraries are installed without errors.
- Check that the imagenet_labels.txt file is in the correct directory.

5 Product Documentation

All code was written in Python.

5.1 Object Recognition

5.1.1 BOM (Bill of Materials)

N/A

5.1.2 Equipment list

- PC that can run an IDE
- Webcam connected to said PC

5.1.3 Instructions

1. Install the Required Libraries
 - Make sure you have the required Python libraries installed. Run the following commands in your terminal or command prompt:
 - `pip install opencv-python`
 - `pip install numpy`
 - `pip install tensorflow`
2. Download the imagenet_labels.txt File
 - a. Download the ImageNet labels file and save it in the same directory as the Python script.
3. Save the Python Code
 - Copy the Python code provided above.

- Save it in a file named `object_recognition.py` in the same directory as the `imagenet_labels.txt` file.
4. Run the Python Script
- Open a terminal or command prompt.
 - Navigate to the directory where you saved the script.
 - Run the script with the following command:
 - `python object_recognition.py`
5. Interact with the Program
- The script will open your webcam.
 - It will detect objects in real time, displaying the object label and confidence percentage on the video feed.
 - To exit the program, press 'q' on your keyboard.
- 6. Troubleshooting**

If the program doesn't work:

- Ensure that your webcam is properly connected and working.
- Verify that all required libraries are installed without errors.
- Check that the `imagenet_labels.txt` file is in the correct directory.

5.2 Testing & Validation

- **Testing Focus:** Object recognition accuracy.
- **Issue:** Early tests misidentified objects (e.g., mistaking a water bottle for a lighter).
- **Fix:** Adjusted the code to improve object recognition.

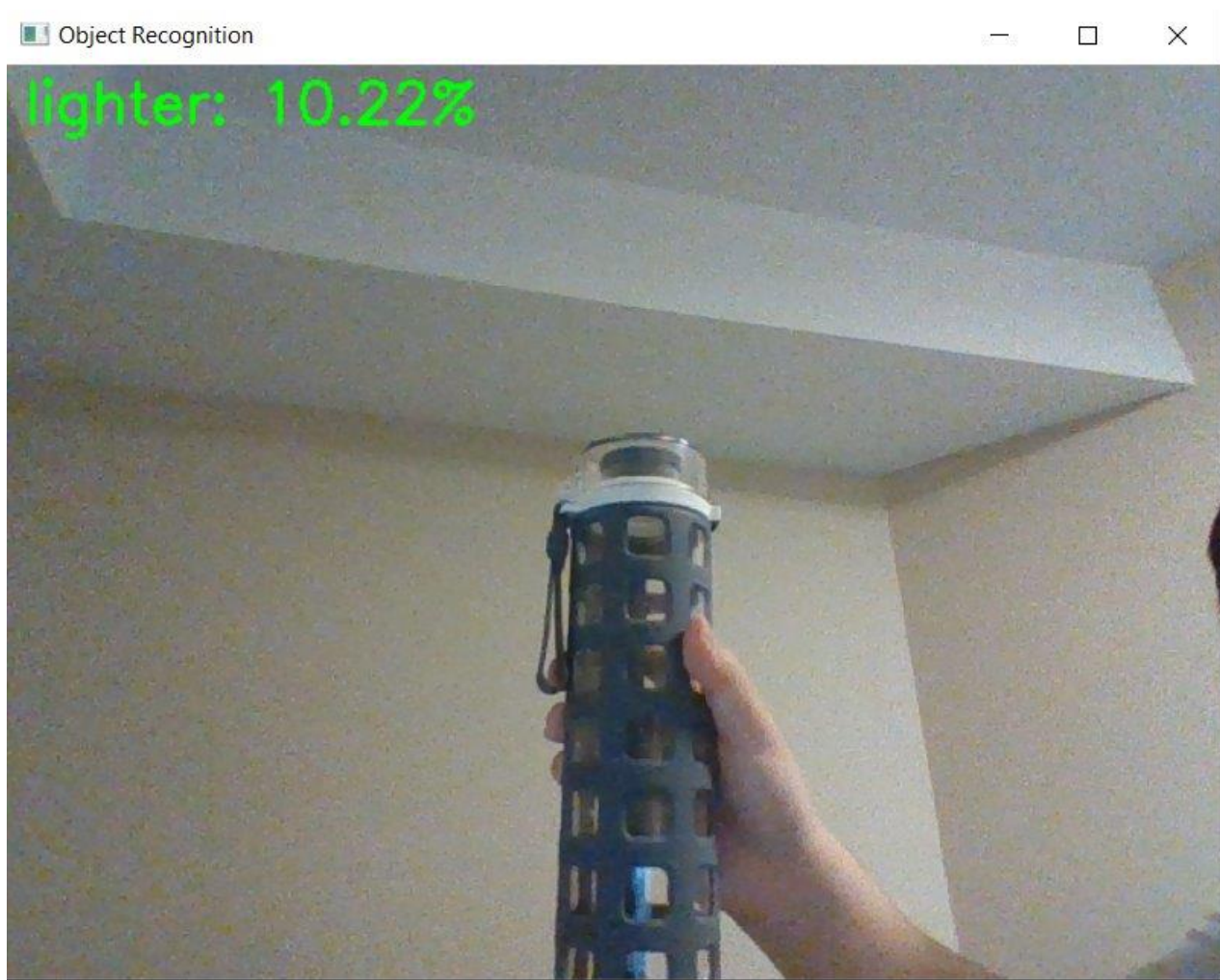


Figure 4. Screenshot of software misidentifying object.

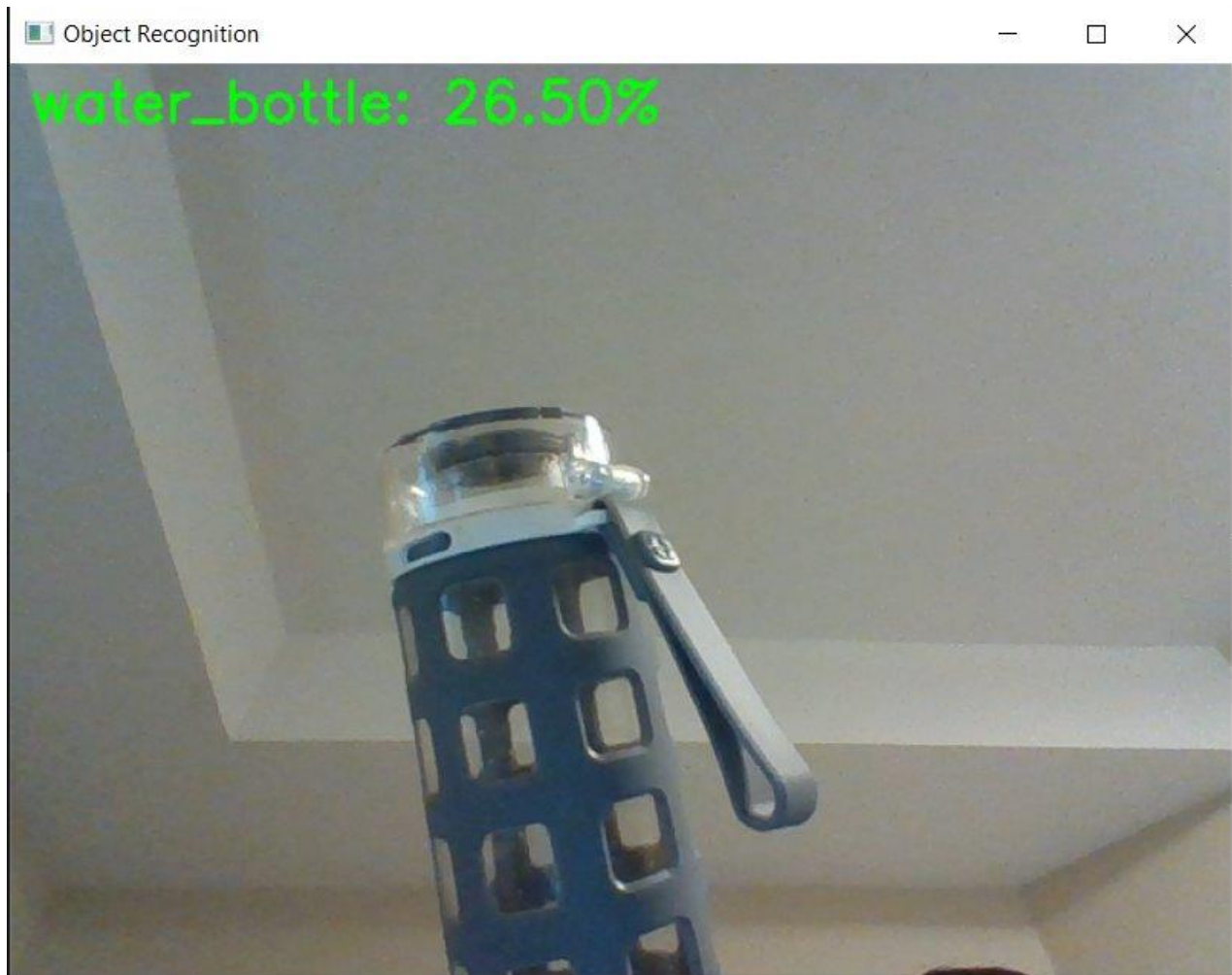


Figure 5. Same screenshot as Figure 3.

6 Conclusions and Recommendations for Future Work

We learned how to prototype a software project and use APIs. We would continue to develop this prototype if we had more time. We had higher hopes for this project, such as adding voice commands and a GPS, but due to time constraints, we had to cut these additional features.

3 Bibliography

Eyeglasses.com. (n.d.). *39 glasses statistics*. Vision Magazine. Retrieved December 3, 2024, from <https://blog.eyeglasses.com/vision-magazine/glasses-statistics/>

APPENDICES

7 APPENDIX I: Design Files

Below is the file on MakerRepo. The file name is “ShabodiSmartGlasses”.

[GlassTek Group 23 | MakerRepo](https://makerepo.com/Skylar/2367.glasstek-group-23)

Table 3. Referenced Documents

Document Name	Document Location and/or URL	Issuance Date
ShabodiSmartGlasses	https://makerepo.com/Skylar/2367.glasstek-group-23	Dec 3, 2024

8 APPENDIX II: Other Appendices

N/A