



MOD002690

Final Year Project

Mobile Gesture and Object Detection using TensorFlow

SID: 1920333/1

Abstract

This project studies the object and hand gesture detection of mobile application implemented to smartphone for different type of users. As a result, there are mobile application with the similar functionalities in the market, but it is not as viral as the other type of mobile application. It will probably because of there is no focus of AI detections implementing into smartphone in order to use detection. Most of the time, object detection can be seen in car censor, video surveillances, crowd counting and many more while hand gesture detection are mostly be used in changing TV channel, volume increment and decrement, etc. This proposed application is developed using TensorFlow with Android operating system and being tested out.

Keywords

Hand Gesture Detection, Double Tap Detection, Object Detection, Object Recognition, Classification

Acknowledgement

A special thanks to the supervisor of this study, Ms Vasugi, where she provides guidance and advice throughout the whole project while carried out from beginning until the end of the research. Specially thanks to friends and those who helps in this study. It is helpful and provided support for this study to be completed within time and success with it.

Table of Contents

Abstract	2
Keywords.....	2
Acknowledgement.....	2
1. Introduction.....	4
1.1. Aims and Objectives of the study.....	4
1.12. Literature Review	5
2. Methodology	9
2.1. Methodology Used	9
2.2. Proposed Application.....	10
2.3. Proposed Design.....	11
2.4. Project Development Tools	13
2.5. Application Pseudocode.....	14
2.5.1. Swipe Gesture detection module	14
2.5.1. Double Tap Gesture detection module.....	15
2.5.3. Object Detection module	15
2.5.4. Code snippet and description	16
3. Implementation and Result	19
3.1. Flow Chart.....	21
3.2. Test Plan.....	22
3.3. How to use.....	23
4. Discussion.....	24
5. Future Work	27
6. Conclusion	29
7. References.....	30
8. Appendix	32

1. Introduction

The uses of electrical technology such as smartphone, television, and censors are getting important year by year and these technologies consist of many techniques to be implemented in order to make our live easier. Artificial Intelligence (AI) also played an important role in bringing more advance features and easiness to the people nowadays. AI technology can be seen everyway nowadays such as chatbot, robotic arms, voice assistance, weather estimation and many more. In this project, object detection and hand gesture detection also using the AI technology that implemented in TensorFlow.

1.1. Aims and Objectives of the study

The aim for this project is to evaluate the existing of smartphone with object and hand gesture detection features. There are many mobile applications nowadays has been reviewed to identify the suitability and usefulness of the features so that the featured mobile application can be developed.

In order to make it happens, there are two main features that will be implemented, which are the object identification detection with reading ability and gesture recognition detection with mobile functionalities into smartphone as an operating system. In fact, in this paper project, there are only mobile application created instead of operating system as the limitation of software and time uses during development. After doing some research on this related fields, these features do not come in one mobile application but one specific functionality in one mobile application. For instant, there are mobile application that implement only object detection within screen and display or read the name.

Meanwhile, there are no existing mobile application that capture human hand gesture and perform some or specific action in the smartphone. Hence, there will be very less button involved or even no button in the application as the actions are mostly be done using hand gesture and object detection implemented to every pages. At the same time, this application developed for different type of user, no matter the user is an adult, a kid, or even the user is a blind or a deaf! Hence, after the development of this proposed features in a single application, there will be comparations of the developed and existing mobile application with similar functionalities towards the target audience and the effectiveness itself.

1.12. Literature Review

The aim of machine learning and computer vision is to incorporate human skills for data sensing, data interpretation, and action taking based on historical and current results into computers. The field of machine learning and computer vision is still growing. The Internet of Things, the Industrial Internet of Things, and brain human interfaces all involve computer vision. Machine learning and computer vision are used to understand and track dynamic human behaviours in multimedia sources (Asharul, 2019). By having the computer to understand something from somewhere else, image processing plays a major role in detecting the segments and specification of an object from either image or video. Computer Vision's main goal is to generate templates, data extracts, and details from images, while Image Processing is concerned with applying computational transformations to images, such as sharpening and contrast, among other things. At the same time, computer vision operates by stimulating human visualisation using an algorithm and optical sensors to retrieve useful information from an object automatically (Victor, 2018). Machine learning can deliver powerful methods for automating the acquisition of visual models, adapting task parameters and representation, converting signals to symbols, constructing trainable image processing systems, concentrating attention on the target object, and learning when to implement which algorithm in a vision system from the viewpoint of computer vision systems. Meanwhile, computer vision can bring fascinating and difficult problems in the context of machine learning systems. In computer vision applications, machine learning algorithms can be used in at least two ways, it is to enhance understanding of the surrounding world, and to improve the translation of perceived signals into internal representations and to bridge the distance between the system's internal representations of the world and the information it requires to complete its mission (N. SEBE, 2005).

Humans can automatically tell what objects are in a picture, where they are, and how they behave only by looking at them. The human visual system is fast and precise, enabling us to execute complicated activities like driving without much thought. A fast and accurate object detection technique will enable computers to drive cars without specialized sensors, assistive devices to communicate real-time scenario information to human users, and general-purpose, sensitive robotic systems to emerge (Joseph Redmon, 2016). Object recognition systems use object templates that are known in advance to find objects in the real world (Sharma, 2017). We should not only focus on classifying various images to gain a full

understanding of them, but also attempt to accurately approximate the concepts and positions of objects found in each image to gain a complete understanding of them. Object detection is the name given to this mission, which is made up of subtasks like face detection, pedestrian detection, and skeleton detection (Xindong Wu, 2019).

In computer vision, object recognition is a key challenge. Many detection pipelines begin by extracting a series of robust features from the input images (Haar, SIFT, HOG, convolutional features). Then, to classify objects in the function space, classifiers or localizers are used (Santosh, 2016). Object detection is one of the most basic computer vision issues, and it can provide useful knowledge for semantic interpretation of images and videos. It is used in a number of applications, including image classification, human behaviour analysis, facial recognition, and autonomous driving (Zhao, 2019). To detect an object in an image or video, the system requires a few components, including a model database, a function detector, a hypothesiser, and a hypothesiser verifier (Nileshsingh, 2017). Since various objects can be appeared in different parts of the image and have different aspect ratios or sizes, scanning the whole image with a multi-scale sliding window is a natural option.

Hand gestures are a form of nonverbal communication that can be extended to a few areas, including deaf-mute communication, robot control, human-computer interaction (HCI), home automation, and medical applications. Many various methods for detecting hand movements have been used, including those relying on instrumented sensor technology and computer vision (Munir Oudah, 2020). Because of wide variety of applications and ability to effectively communicate with machines through human-computer interaction, the hand gesture recognition technology has gotten a lot of attention in recent years (Rafiqul, 2012). Human action recognition, posture prediction, gesture recognition, and hand gesture recognition are among the many human-computer research exercises in computer vision and machine learning that are especially important due to their numerous possible applications. Many computer vision applications, such as human-computer interaction, sign language identification, hand gesture analysis, driver hand movement tracking, and virtual reality, depend on precise hand gesture detection and recognition in cluttered environments (Adam Ahmed, 2019). Gestures may be static, such as a stance or a specific position, which requires less computational complexity, or dynamic, such as a series of postures, which is more complicated but suitable for real-time situations (Noor, 2012).

Since it allows for contactless communication between humans and computers, the camera vision-based sensor is a common, appropriate, and applicable technique. However, this method faces several difficulties, including lighting variations, background problems, the effect of occlusions, dynamic backgrounds, rendering time against resolution and frame rate, and foreground or background subjects of the same skin tone or otherwise resembling a hand (Ali, 2020). Hand gestures have the inherent capacity to easily reflect ideas and behaviours, using these various hand forms, which are recognised by gesture recognition systems and translated to produce subsequent events, has the potential to offer a more natural interface to the computer system. Given skin tones and texture differ so rapidly from person to person and continent to continent, the outcome of hand gesture identification can vary. Additionally, colour texture varies under varying lighting conditions, resulting in changes in observed effects. A vision-based hand gesture recognition system that operates on shape-based functionality for hand gesture recognition is recommended for using different hand movements to facilitate real-time use. It is a fundamental fact because, under normal situations, everyone has the same hand form with one thumb and four fingers (Panwar).

Kartik Umesh Sharma (2017) et al. suggested an object recognition method that detects real-world objects in a visual image or video, where the object may be from any type of objects, such as people, vehicles, and so on. Due to routine shifts in object motion and variance in scene size, occlusions, presence variations, and ego-action and illumination adjustments, Mukesh Tiwari (2017) et al. presented object identification and tracking as one of the key areas of study. Feature collection, in particular, is critical in object tracking. It's used in a variety of real-time applications, such as car vision and video monitoring. To solve the problem of identification, monitoring of object orientation and presence is used. The monitoring algorithm is at the heart of the algorithm, which smooths out the video chain. In the other hand, only a few approaches make use of previously collected data such as object type, colour, texture, and so on. Karanbir Chahal and colleagues (2018) suggested Object recognition is the process of identifying, localising, and classifying an object in an image. It has a broad range of uses and is an essential part of vision-based computing systems.

The aim, according to Geethapriya. S (2019) et al., is to detect objects using the You Only Look Once (YOLO) approach. As opposed to other target detection algorithms, this approach has many benefits. Other algorithms, such as Convolutional Neural Network and Fast Convolutional Neural Network, do not look at the image fully, whereas YOLO looks at the image completely by estimating the bounding boxes with a convolutional network and the class

probabilities for these boxes, and detects the image faster than other algorithms. Guo et al. (2012) proposed a method for detecting objects in video frames called object recognition. The simulation results show that this technique is efficient, precise, and durable for detecting generic object classes with good efficiency. The emphasis should also be on improving classification accuracy in real-time object recognition.

2. Methodology

2.1. Methodology Used

There are qualitative and quantitative research methodology used in this report where the qualitative research focuses on the information by doing observation, study and analyses previous existing research to obtain ideas and findings, while the quantitative research is focusing throughout the paper during development where testing and analytics will be carried out.

Agile Software Development has been used throughout the project as it is one of the simplest and effective processes during the whole software development. Continuous preparation, research, progress, team engagement, evolutionary development, and early implementation are all concepts used to represent agile software development methodology.



The figure above showing the whole process taken part in this research project. The activities that must be done and in what order of priority they must be performed are very beneficial during production due to a well-defined collection of criteria in Agile Development Methodology. Predicting risks and devising successful contingency strategies becomes easier with improved exposure. There are more opportunities to recognise and forecast risks in the Agile framework, as well as prepare and ensure that the project runs smoothly.

During the process of development, a research has been done where the statistic of usage of Artificial Intelligence (AI) are becoming popular and important year by year. Artificial Intelligence involving a lot of industry around the world nowadays no matter it is in medical, car, robot or even our daily home appliances.

The object detection system basically comprises of two main phases namely, which are the learning and the testing phase. The learning process is primarily for the classifier to recognise the objects in the picture that is provided as feedback to the machine. The learning process is further divided into two categories: learning through instruction and learning through validation. Learning by training consists mostly of the learning block, which defines a proper learning system, which can be part-based or patch-based, for example. The testing phase's key goal is to determine whether an entity is present in the picture that the device receives as feedback. If you answered yes, what object class does it belong to?

2.2. Proposed Application

The proposed mobile application is originally developed as an operating system of smartphone. Due to the limited time and software, the idea has been changed into a mobile application to imitate as operating system. The application's main page or home page will be act like the home screen of a smartphone. An "button option-less" idea is being implement into this software, where there will be no button option at main page for user to select and proceed. Meanwhile, user will just have to swipe or tap through the screen to navigate to different specific pages or perform other functions.

There are two main groups of users that this mobile application (represent as operating system) be useful to them in their daily life, and they are the blinds and the learner. It is suitable for blinds because this application can perform functions by just swiping and tapping on the screen to do tasks. There will be four swipe actions that can be done in the main page of the mobile application. The four actions consist of swiping up, down, left, and right. Each of these swipe actions will navigate user to a new page. After navigating to new page, user can either press the back button on the top of the page or just simply double tap on the page screen to go back to the main page. Meanwhile, in the specific page after navigated from main page, user can do the swiping action again in order to have other features to be done at the page.

There are three main features for user to choose out of the four swiping actions as there is another one action that do not have any specific feature that need to be implemented in this proposed mobile application. The three main features involve swiping up (navigate to Settings page), swiping left (navigate to Object Detection page), and swiping right (navigate

to Gesture Detection page). In Settings page, user can select which swiping action navigating to which page and perform what features. In Object Detection page, object will be detected within the camera based on the keyboard or by sound input from user. If there is object detected within the camera matched to the search, the application will play a sound ‘Detected’ and green label will be showed on screen. Else, it will play a sound ‘Not Detected’ and red label will be displayed on screen. At the same time, if user wanted to search a new object, user can long press the screen, the input will be reset, and the phone will vibrate so that the blinds will get to know that he or she can start a new search. On the other page, in Gesture Detection page, it is mainly developed for the blinds. The idea of gesture detection is originally to be developed together with swipe action in the main page. Due to the limitations met in this proposed application during development, the feature has been separated for easy reference. In Gesture Detection page, the camera will detect the hand gesture made by user and the program will perform specific task such as calling specific person. As for who will be called, it can be set in the Settings page. The flow of the application can be referred as the flowchart below.

2.3. Proposed Design

The designs and functions of the mobile application of different pages are present as below.

Page	Description
Main Page	<ul style="list-style-type: none"> • This is the main page of the mobile application. • It can detect different gestures such as swiping up, down, left, and right. • It does not have double tap screen to go back to previous page.
Real Time Object Detection Page	<ul style="list-style-type: none"> • This is the real time camera object detection page. • It can detect different gestures such as swiping up, down, left, and right. • It has double tap screen to go back to previous page.

Select Picture for Object Detection Page	<ul style="list-style-type: none"> • This is the real time camera object detection page. • It can detect different gestures such as swiping up, down, left, and right. • It has double tap screen to go back to previous page.
Picture Selection Page	<ul style="list-style-type: none"> • This is the real time camera object detection page. • It cannot detect different gestures such as swiping up, down, left, and right. • It does not have double tap screen to go back to previous page.
Object Detection for Picture Selected Page	<ul style="list-style-type: none"> • This is the real time camera object detection page. • It can detect different gestures such as swiping up, down, left, and right. • It has double tap screen to go back to previous page. • After selecting picture or photo from local storage or online cloud storage, the system will detect and name the object in the provided picture.

2.4. Project Development Tools

The techniques used in the software development process have the potential to make or break a project. After settling on the target environment and programming language(s), as well as the specifications and end goals, the next step in initiating a software development project is to choose the resources that will be used in the process. It is important to understand the different types of resources available, the advantages they can bring, and the consequences of using them.

1. Android Studio

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems or as a subscription-based service in 2020. It is a replacement for the Eclipse Android Development Tools (E-ADT) as the primary IDE for native Android application development. Android Studio 3.0 or later supports Kotlin and all Java 7 language features and a subset of Java 8 language features that vary by platform version.

2. Flutter

Flutter is an open-source UI software development kit created by Google. It is used to develop applications for Android, iOS, Linux, Mac, Windows, Google Fuchsia, and the web from a single codebase. Flutter apps are written in the Dart language and make use of many of the language's more advanced features.

3. TensorFlow

TensorFlow is a free and open-source software library for machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. TensorFlow is a symbolic math library based on dataflow and differentiable programming. It is used for both research and production at Google. TensorFlow was developed by the Google Brain team for internal Google use. Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

4. Metadata

Model descriptions can be defined using TensorFlow Lite metadata. The metadata is a valuable source of information about the model's operations and input or output results.

The metadata is made up of both human-readable sections that express best practises for using the model and machine-readable parts that can be used by code generators including TensorFlow Lite Android and the Android Studio ML Binding functionality.

2.5. Application Pseudocode

There is no server or database implemented in this project. Instead of server and database, metadata is used to read the data as database such as MySQL and Firebase. TensorFlow Lite metadata provides a standard for model descriptions. The metadata is a valuable source of information about the model's operation and its input and output data. The metadata consists of both parts that are understandable by humans and share the best practices for using the model and Code generators, such as the TensorFlow Lite Android code generator and the Android Studio ML Binding feature, can use machine-readable bits. There are multiple models that can be used in metadata prepared by Tensorflow via this link: https://www.tensorflow.org/lite/guide/hosted_models#object_detection. The complete version of the mobile application together with the metadata can be referred to attached project file and in Appendix section.

2.5.1. Swipe Gesture detection module

The mobile application starts by swiping gesture on screen by user. Meanwhile, there are swipe gesture detection in every screen. Each of the swipe action perform different functions and tasks given by user. Logically, the tasks can be set and given by user through the settings page. In this application, there are only 4 different swipe actions can be done in every page which are by swiping up, down, left, and right. These are the swipe actions that can be detected easily instead of swiping at 45° , 135° , 225° , 315° , etc. The following is a simple version of pseudocode of the process.

IF swiped up

DIRECT to ... page

ELSE IF swiped down

DIRECT to ... page

ELSE IF swiped left

DIRECT to ... page

ELSE IF swiped right

DIRECT to ... page

ENDIF

2.5.1. Double Tap Gesture detection module

Besides having swipe gesture detection in every page, double tap gesture also implemented into the application but not for the first page. The main purpose of having double tap gesture detection is to allow user to navigate to previous page by tapping twice on the smartphone's screen within 1.5 seconds.

IF double tapped

NAVIGATE BACK to ... page

ENDIF

2.5.3. Object Detection module

There are two ways of the object being detected in this system. Firstly, user can choose to use a real time camera to detect anything within the camera sight. User do not have to press any button to do it. Next, user can upload a photo or picture from local phone storage or from online cloud storage for the system to identify the object in the picture given. The system will identify and name the object in the picture for the user.

IF real time camera

SWITCH ON camera and detect

ELSE IF upload picture

OPEN local album / online cloud storage

SELECT picture chosen

DETECT the object from picture

2.5.4. Code snippet and description

Code	Description
<pre>Expanded(child: SwipeDetector(onSwipeUp: () { print("Main Swiped Up"); }, onSwipeDown: () { print("Main Swiped Down"); }, onSwipeLeft: () { print("Main Swiped Left"); Navigator.push(context, MaterialPageRoute(builder: (context)=> StaticImage(),)); }, onSwipeRight: () { print("Main Swiped Right"); Navigator.push(context, MaterialPageRoute(builder: (context)=> LiveFeed (cameras),)); },)),</pre>	<ul style="list-style-type: none"> This is where the system detects the swipe actions to be done by user in order allow the system to navigate user to new page or do some tasks. In this proposed system, it swipes left navigate user to upload picture to do object detection page, and swipe right to navigate user to real time camera object detection page.

<pre>Expanded(child: SwipeDetector(onSwipeUp: () { print("Swiped Up"); }, onSwipeDown: () { print("Swiped Down"); Navigator.pop(context); }, onSwipeLeft: () { print("Swiped Left"); // ignore: unnecessary_statements getImageFromCamera(); }, onSwipeRight: () { print("Swiped Right"); // ignore: unnecessary_statements getImageFromGallery(); },)),</pre>	<ul style="list-style-type: none"> This is where the system detects the swipe actions to be done by user in order allow the system to navigate user to new page or do some tasks. In this proposed system, it swipes left to allow user take photo at the moment he wants to do object detection, it swipes right to allow user select picture or photo from local storage or online cloud storage to do object detection.
---	--

```

if (_recognitions == null) return [];
if (_imageHeight == null || _imageWidth == null)
return [];

double factorX = screen.width;
double factorY = _imageHeight / _imageWidth *
screen.width;
Color blue = Color.fromRGBO(37, 213, 253, 1.0);

return _recognitions.map((re) {
return Positioned(
  left: re["rect"]["x"] * factorX,
  top: re["rect"]["y"] * factorY,
  width: re["rect"]["w"] * factorX,
  height: re["rect"]["h"] * factorY,
  child: Container(
    decoration: BoxDecoration(
      borderRadius:
BorderRadius.all(Radius.circular(8.0)),
      border: Border.all(
        color: blue,
        width: 2,
      ),
    ),
    child: Text(
      "${re["detectedClass"]}"
      "${(re["confidenceInClass"] *
100).toStringAsFixed(0)}%",
      style: TextStyle(
        background: Paint()..color = blue,
        color: Colors.white,
        fontSize: 12.0,
        print((re["confidenceInClass"] *
100).toStringAsFixed(0))
      ),
    ),
  );
});
}).toList();
}

```

- This is where the system recognizes the object from the image no matter it is using real time camera detection of through picture.
- After the system recognized the object, it will then draw a box outside the object and name it together with percentage of similarities under the box.

```

@required String path,
String model = "SSDMobileNet",
double imageMean = 127.5,
double imageStd = 127.5,
double threshold = 0.1,
int numResultsPerClass = 5,
// Used in YOLO only
List anchors = anchors,
int blockSize = 32,

```

- This is where the system recognizes the object from the image no matter it is using real time camera detection of through picture.

```

int numBoxesPerBlock = 5,
bool asynch = true,
}) async {
return await _channel.invokeMethod(
'detectObjectOnImage',
{
"path": path,
"model": model,
"imageMean": imageMean,
"imageStd": imageStd,
"threshold": threshold,
"numResultsPerClass": numResultsPerClass,
"anchors": anchors,
"blockSize": blockSize,
"numBoxesPerBlock": numBoxesPerBlock,
"asynch": asynch,
},
);
}

```

- It identifies the class and size of the objects within the sight in order to do comparison and categorize them.

```

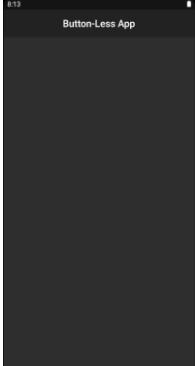
static Future<String> loadModel(
{@required String model,
String labels = "",
int numThreads = 1,
bool isAsset = true,
bool useGpuDelegate = false}) async {
return await _channel.invokeMethod(
'loadModel',
{
"model": model,
"labels": labels,
"numThreads": numThreads,
"isAsset": isAsset,
'useGpuDelegate': useGpuDelegate
},
);
}

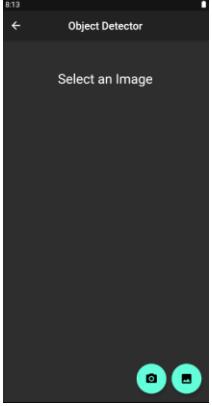
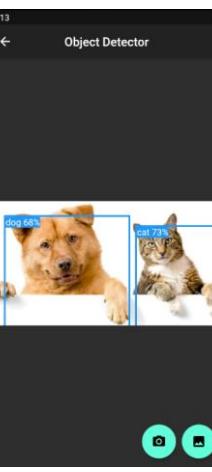
```

- This is where the system recognizes the object from the image no matter it is using real time camera detection or through picture.
- After getting the information about the image in binary form, they will be used to compare the data and information in metadata.
- It loads given metadata file to finds the most suitable and similarities of the object detected with the data given.

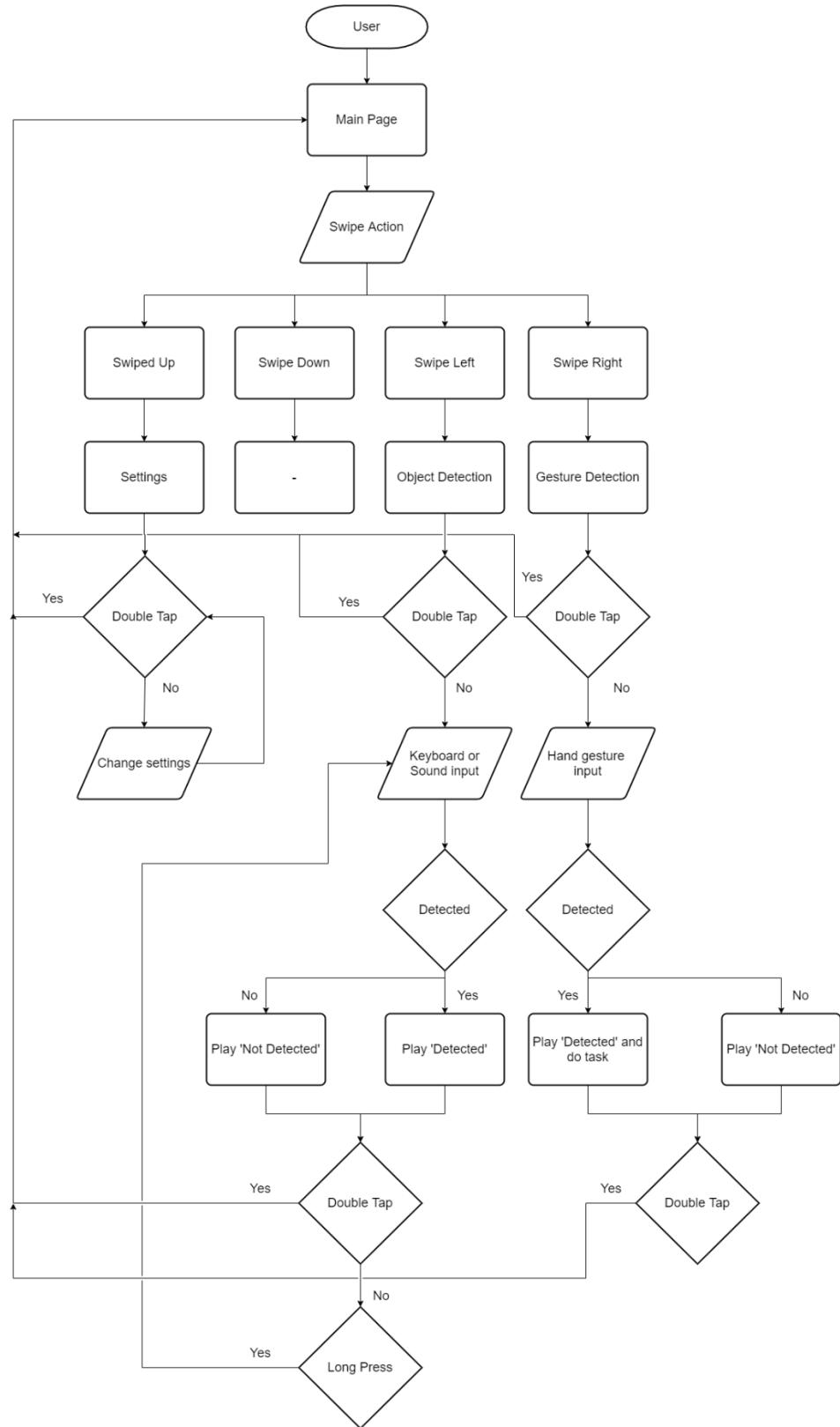
3. Implementation and Result

The mobile application has been developed using Android operating system and it is being tested using real time camera or image from phone album. This is an application that use gesture detection as main functions and actions triggering method. Hence, there will be very less or even no button involved in the application. At the same time, the Graphical User Interface (GUI) in this application will not as classy or beautiful as the other application out there. The following are the screenshot before and after the detection.

Screenshot	Description
	<ul style="list-style-type: none"> This is the main page of the mobile application. It can detect different gestures such as swiping up, down, left, and right. It does not have double tap screen to go back to previous page. In this proposed system, it swipes left navigate user to upload picture to do object detection page, and swipe right to navigate user to real time camera object detection page.
	<ul style="list-style-type: none"> This is the real time camera object detection page. It can detect different gestures such as swiping up, down, left, and right. It has double tap screen to go back to previous page. In future work, it swipes left allow system to read the object name(s) detected within the camera, and swipe right to display all the name of the detected object in words in a GUI container.
	<ul style="list-style-type: none"> This is the real time camera object detection page. It can detect different gestures such as swiping up, down, left, and right. It has double tap screen to go back to previous page.

	<ul style="list-style-type: none"> In this proposed system, it swipes left to allow user take photo at the moment he wants to do object detection, it swipes right to allow user select picture or photo from local storage or online cloud storage to do object detection.
	<ul style="list-style-type: none"> This is the real time camera object detection page. It cannot detect different gestures such as swiping up, down, left, and right. It does not have double tap screen to go back to previous page. This is where user will have to upload their picture or photo they wished to do object detection.
	<ul style="list-style-type: none"> This is the real time camera object detection page. It can detect different gestures such as swiping up, down, left, and right. It has double tap screen to go back to previous page. After selecting picture or photo from local storage or online cloud storage, the system will detect and name the object in the provided picture. In this proposed system, it swipes left to allow user take photo at the moment he wants to do object detection, it swipes right to allow user select picture or photo from local storage or online cloud storage to do object detection.

3.1. Flow Chart



3.2. Test Plan

No.	Implementation	Expected Result	Result
1	Page Navigation using button.	User will be directed to a new page.	Pass
2	Go back to previous page using button.	User will be go back one page before.	Pass
3	Implement Swipe Gesture Detection for different page navigation and remove button to replace them.	User able to swipe up, down, left and right to navigate to different pages.	Pass
4	Implement Double Tap Gesture Detection to go back to previous page and remove button to replace them.	User able to double tap on screen and directed one page back before the current page.	Pass
5	Implement real time camera detection and identify object scanned within scope / sight.	User able to directly scan the sight using real time camera function to identify the objects within the camera without needed to press any button.	Pass
6	Implement action for user to upload picture from album or cloud storage in order to test and identify objects within the picture uploaded.	User able to upload picture from local phone album or from online cloud storage in order to scan and identify the objects within the uploaded picture.	Pass

3.3. How to use

There are two main groups of users that this mobile application (represent as operating system) be useful to them in their daily life, and they are the blinds and normal people like us. It is also suitable for blinds because this application can perform functions by just swiping and tapping on the screen to do tasks. There will be four swipe actions that can be done in every page of the mobile application. The four actions consist of swiping up, down, left, and right. Each of these swipe actions will navigate user to a new page. After navigating to new page, user can either press the back button on the top of the page or just simply double tap on the page screen to go back to the main page. Meanwhile, in the specific page after navigated from main page, user can do the swiping action again in order to have other features to be done at the page.

There are three main features for user to choose out of the four swiping actions as there is another one action that do not have any specific feature that need to be implemented in this proposed mobile application. The three main features involve swiping up (navigate to Settings page), swiping left (navigate to Object Detection page), and swiping right (navigate to Gesture Detection page). In Settings page, user can select which swiping action navigating to which page and perform what features.

In Object Detection page, object will be detected within the camera based on the keyboard or by sound input from user. If there is object detected within the camera matched to the search, the application will play a sound ‘Detected’ and green label will be showed on screen. Else, it will play a sound ‘Not Detected’ and red label will be displayed on screen. At the same time, if user wanted to search a new object, user can long press the screen, the input will be reset, and the phone will vibrate so that the blinds will get to know that he or she can start a new search. On the other page, in Gesture Detection page, it is mainly developed for the blinds. The idea of gesture detection is originally to be developed together with swipe action in the main page. Due to the limitations met in this proposed application during development, the feature has been separated for easy reference. In Gesture Detection page, the camera will detect the hand gesture made by user and the program will perform specific task such as calling specific person.

4. Discussion

The techniques used in the software development process are very important as they have the potential to make or break a project. After settling on the target environment and programming language(s), as well as the specifications and end goals, the next step in initiating a software development project is to choose the resources that will be used in the process. It is important to understand the different types of resources available, the advantages they can bring, and the consequences of using them.

The aim of object detection is to find all instances of a recognised class of objects in an image or video, such as humans, cars, or faces. Object identification is problematic due to environmental lighting changes, sudden changes in target appearance, contrasting non-target features in the background, and occlusions. The system of object detection employs semiautomatic or automatic detection techniques.

This is a system where supposedly developed as an operating system. Due to the shortage and limitation of time, it has been developed into a single mobile application using Flutter development tools in Android Studio Integrated Development Environment (IDE). The operating system used for this development is Android operating system and an emulator of device with version Android 11 has been used to test the system throughout the whole process. Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems or as a subscription-based service in 2020. It is a replacement for the Eclipse Android Development Tools (E-ADT) as the primary IDE for native Android application development. Android Studio 3.0 or later supports Kotlin and all Java 7 language features and a subset of Java 8 language features that vary by platform version. As for flutter it is an open-source UI software development kit created by Google. It is used to develop applications for Android, iOS, Linux, Mac, Windows, Google Fuchsia, and the web from a single codebase. Flutter apps are written in the Dart language and make use of many of the language's more advanced features.

The hand gesture and double tap detection are implemented using the plugins prepared in Flutter development tools library in order to detect the actions done by user on the screen

before continuing. In the future, the finger hand gesture detection can be added to create a more accessible system for user to carry more tasks.

The object detection in this proposed system can be carried through two different methods. This is because there are two main target audience or user for this system which are the blinds and normal people like us. The two different methods include real time camera object detection and object detection from picture. It is obvious that a blind cannot click on a button to select a picture and upload for the system to do detection. Hence, this is where the real time is useful for them as the blinds can just swipe on the screen and it will directly bring to the page and instantly detect the sight using real time camera without pressing anything. At the same time, the system will read the object name that has been detected in the real time camera. As for the picture upload, user will just have to choose the picture or photo taken and upload to the system. The system will identify and name out the object detected within the picture provided by user.

So, how is this all work overall? The object detection method that implemented in this proposed system are using the metadata provided in Tensorflow. Tensorflow had been used to read the metadata file for further detection in the system. Model descriptions can be defined using TensorFlow Lite metadata. The metadata is a valuable source of information about the model's operations and input or output results. It is made up of both human-readable sections that express best practices for using the model and machine-readable parts that can be used by code generators including TensorFlow Lite Android and the Android Studio ML Binding functionality. TensorFlow is a free and open-source software library for machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. TensorFlow is a symbolic math library based on dataflow and differentiable programming. It is used for both research and production at Google. TensorFlow was developed by the Google Brain team for internal Google use. Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

For classification, neural networks are very common. Classification is one of the most important principles of object detection. The model takes an image as input and generates a potential object type based on how it was taught. Its efficiency is determined by the network infrastructure used and how well the network has been educated. Rotation, scale, subtly angled, and even partial occlusion have little effect on the Support Vector Machine. Template matching

is quick and simple to add new classes to, but it is computationally expensive when the geometry transform is complex. Its response rate can be increased if it is paired with a genetic algorithm. Furthermore, the hybrid form does not necessitate the meticulous preparation of several models. In real-world image sequences, different moving regions refer to different moving objects. Since it is important to accurately differentiate them from other moving objects for subsequent tasks such as monitoring and movement recognition, the desired moving object area should be distinguished from other moving object regions. About any object detection programme is concerned with the possibility that a detected object belongs to the proposed class. A chance score is normally allocated to each area defined by an object detection algorithm. The cumulative likelihood that a proposed item is an individual will be calculated as the average of these ratings, as determined by different methods.

Since static image has been also used in object detection in this proposed system, it also uses Background Subtraction Method to get accurate reading and classification. For static backgrounds, the Context Subtraction approach is appropriate. The identification of an object in a video series allows the pixels to be classified as foreground or background. A context model is a representation of a scene in the object detection process. Context modelling is one of the most important and difficult aspects of background subtraction.

5. Future Work

After implemented with multiple features and actions into the mobile application as an operating system, there are still a lot of space for us to improve them. This is due to the shortage and limit of time to develop the whole system, there are only important, and some functions added to current project. Hence, there will still be many improvements that can be done in the future.

First of all, in this project, the system has been implemented in mobile application form. The main target for this system is to develop it into an operating system similar to Android and iOS. By develop current system into an operating system, it can be act as a smartphone start from zero without any button or very less require button for it to be used. Meanwhile, almost all the actions and tasks in the smartphone are done using hand gesture and object detection. Hence, with this “only detection” system, user do not have to look into the device to locate where the button is and press on it. They can just swipe through the screen even closing their eyes! User will just have to know what and where the swipe action goes to. For example, while the phone is sleep, user will just have to swipe up the screen and the phone will immediately power on and open the contact page for user.

Besides, read and speak function can be added to the system. For example, iOS has been implemented a function called as ‘Voice Over’ where it will tell user what the user has done, such as telling user that he or she pressed on the Settings application. But in this system, there will some modification of this features where the speak function can be toggle after the camera detected the objects on screen and it will read and tell the user the object name that has been detected. At the same time, there will be option for user to select if they want the system to read for them as this system are develop for the blinds and normal user. If the read mode is on, it will be talking almost the same as while we are using the Global Positioning System (GPS). It will tell user whatever within the camera. At the same time, it can be use in learning session where user can locate and name the object, then allow the system to pronounce the name of the object after detection.

Next, customization of action can be done by user by adding some settings and adjustment into the Settings page. By doing this, user can select what to do based on which action be done in each of the page. At the same time, the third-party mobile application can also be adjusted in the Settings page in order for user to have more choices of application to be

linked. As it is involving the origin and third-party application, the settings have to be dynamic so that the system can manage and give orders to the application smoothly. For example, the Facebook application do not come together in the devices, but the user wanted to open the Facebook by swiping up the screen. With this, the user can go to Settings page and select the action of the swipe up features, then choose the Facebook application as the action to be perform while user swiping up the phone's screen.

In future, developer can create their own metadata file for the system to do classification and other stuff. A metadata is a valuable source of information about the model's operations and input or output results. The metadata is made up of both human-readable sections that express best practices for using the model and machine-readable parts that can be used by code generators including TensorFlow Lite Android and the Android Studio ML Binding functionality. By creating own metadata, developer can copy the existing information from another metadata file and add their own data into it. This will make the metadata file to be more accurate while doing comparison and classification during detection.

Finally, it is important for the detection to have fast and accurate reply. In this proposed system, the real time camera of object detection cannot identify all object with the correct classification, category and name. For example, it will some time identify a water bottle as television. The main and important speciality in this system is the hand gesture and object detection. Hence, it is important to have at least these feature and function can be implemented and working fine so that it would not cause any other problem to other features in the future.

6. Conclusion

As a conclusion, it is possible for different detection implemented into one single system but some of the cautions have to be taken. The cautions include the speed and performance of the device, the battery usage and draining of the device and also the metadata connection for detection database. To create a useable and high reliability of a system, these cautions have to be taken so that the aim of the system can be achieved, and user can use it with ease. In short, it is a system application where it requires very minimum amount of button to function or even no button to carry out tasks and the target audience are for the blinds and normal people. And most importantly, it is a system made up of almost all detection but only hand gesture and object detection in this proposed system as the main operating system of the device from beginning for user to use. When you successfully identify an individual in an image or video, you've created an algorithm that combines object identification and image classification. The technology that allows you to detect artefacts in image data differs from the widely used visual classification methods in many industries.

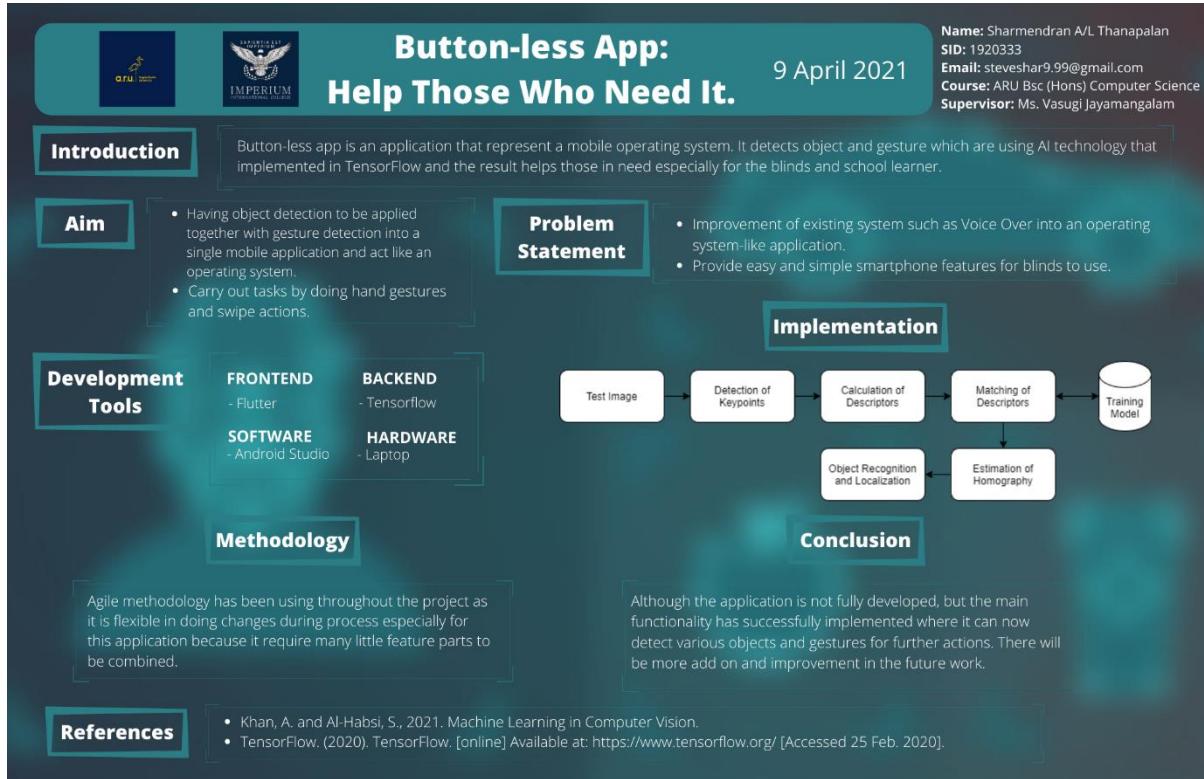
7. References

- TensorFlow. (2020). *TensorFlow*. [online] Available at: <https://www.tensorflow.org/> [Accessed 25 Feb. 2020].
- Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mane, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viegas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. and Zheng, X., 2016. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. [online] Available at: <<http://arxiv.org/abs/1603.04467>>.
- Rafiqul Zaman Khan, Noor Adnan Ibraheem, July 2012. HAND GESTURE RECOGNITION: A LITERATURE REVIEW: International Journal of Artificial Intelligence & Applications (IJAIA), Vol.3, No.4
- Munir Oudah, Ali Al-Naji, and Javaan Chahl, 2020. Hand Gesture Recognition Based on Computer Vision: A Review of Techniques.
- Zhi-hua Chen, Jung-Tae Kim, Jianning Liang, Jing Zhang, and Yu-Bo Yuan, 2014. Real-Time Hand Gesture Recognition Using Finger Segmentation.
- Hsiang-Yueh Lai, Hao-Yuan Ke, and Yu-Chun Hsu, 2018. Real-time Hand Gesture Recognition System and Application: Sensors and Materials, Vol. 30, No. 4 (2018) 869–884.
- Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu, 2019. Object Detection with Deep Learning: A Review.
- Joseph Redmon, Santosh Divvala, Ross Girshick , Ali Farhadi, 2016. You Only Look Once: Unified, Real-Time Object Detection.
- Kartik Umesh Sharma and Nileshsingh V. Thakur, 2017. A review and an approach for object detection in images: Int. J. Computational Vision and Robotics, Vol. 7, Nos. 1/2, 2017
- Juan Wu, Bo Peng, Zhenxiang Huang, and Jietao Xie, 2013. Research on Computer Vision-Based Object Detection and Classification: CCTA 2012, Part I, IFIP AICT 392, pp. 183–188, 2013.

- Adam Ahmed Qaid MOHAMMED , Jiancheng Lv and MD. Sajjatul Islam, 2019. A Deep Learning-Based End-to-End Composite System for Hand Detection and Gesture Recognition.
- Victor Wiley, Thomas Lucas, 2018. Computer Vision and Image Processing: A Paper Review: Vol 2, No 1, June 2018, pp. 28-36.
- Asharul IslamKhan, SalimAl-Habsi, 2020. Machine Learning in Computer Vision: Procedia Computer Science Volume 167, 2020, Pages 1444-1451

8. Appendix

Poster



Main Page

```

import 'package:camera/camera.dart';
import 'package:flutter/material.dart';
import 'package:swipedetector/swipedetector.dart';
import 'package:object_detection realtime/live_camera.dart';
import 'package:object_detection/static%20image/static.dart';
List<CameraDescription> cameras;

Future<void> main() async {
  // initialize the cameras when the app starts
  WidgetsFlutterBinding.ensureInitialized();
  cameras = await availableCameras();
  // running the app
  runApp(
    MaterialApp(
      home: MyApp(),
      debugShowCheckedModeBanner: false,
      theme: ThemeData.dark(),
    )
  );
}
class MyApp extends StatefulWidget {

```

```

@Override
_MyAppState createState() => _MyAppState();
}

class _MyAppState extends State<MyApp> {
@Override
Widget build(BuildContext context) {
return Scaffold(
appBar: AppBar(
title: Text("Button-Less App"),
centerTitle: true,
),
body: Container(
child: Center(
child: Column(
mainAxisAlignment: MainAxisAlignment.center,
children: <Widget>[
Expanded(
child: SwipeDetector(
onSwipeUp: () {
print("Main Swiped Up");
},
onSwipeDown: () {
print("Main Swiped Down");
},
onSwipeLeft: () {
print("Main Swiped Left");
Navigator.push(context, MaterialPageRoute(
builder: (context) => StaticImage(),
));
},
onSwipeRight: () {
print("Main Swiped Right");
Navigator.push(context, MaterialPageRoute(
builder: (context) => LiveFeed(cameras),
));
},
)
],
),
),
);
}
}
}

```

Real Time Camera Object Detection

```

import 'package:camera/camera.dart';
import 'package:flutter/material.dart';

```

```

import 'package:object_detection/realtme/bounding_box.dart';
import 'package:object_detection/realtme/camera.dart';
import 'dart:math' as math;
import 'package:tflite/tflite.dart';

class LiveFeed extends StatefulWidget {
  final List<CameraDescription> cameras;
  LiveFeed(this.cameras);
  @override
  _LiveFeedState createState() => _LiveFeedState();
}

class _LiveFeedState extends State<LiveFeed> {
  List<dynamic> _recognitions;
  int _imageHeight = 0;
  int _imageWidth = 0;
  initCameras() async {

  }
  loadTfModel() async {
    await Tflite.loadModel(
      model: "assets/models/ssd_mobilenet.tflite",
      labels: "assets/models/labels.txt",
    );
  }
  /*
  The set recognitions function assigns the values of recognitions, imageHeight and width to the
  variables defined here as callback
  */
  setRecognitions(recognitions, imageHeight, imageWidth) {
    setState(() {
      _recognitions = recognitions;
      _imageHeight = imageHeight;
      _imageWidth = imageWidth;
    });
  }

  @override
  void initState() {
    super.initState();
    loadTfModel();
  }

  @override
  Widget build(BuildContext context) {
    Size screen = MediaQuery.of(context).size;
    return Scaffold(
      appBar: AppBar(
        title: Text("Real Time Object Detection"),
      ),
      body: Stack(

```

```

children: <Widget>[
    CameraFeed(widget.cameras, setRecognitions),
    BoundingBox(
        _recognitions == null ? [] : _recognitions,
        math.max(_imageHeight, _imageWidth),
        math.min(_imageHeight, _imageWidth),
        screen.height,
        screen.width,
    ),
],
),
);
}
}

import 'package:flutter/material.dart';
import 'package:camera/camera.dart';
import 'package:tflite/tflite.dart';
import 'dart:math' as math;

typedef void Callback(List<dynamic> list, int h, int w);

class CameraFeed extends StatefulWidget {
    final List<CameraDescription> cameras;
    final Callback setRecognitions;
    // The cameraFeed Class takes the cameras list and the setRecognitions
    // function as argument
    CameraFeed(this.cameras, this.setRecognitions);

    @override
    _CameraFeedState createState() => new _CameraFeedState();
}

class _CameraFeedState extends State<CameraFeed> {
    CameraController controller;
    bool isDetecting = false;

    @override
    void initState() {
        super.initState();
        print(widget.cameras);
        if (widget.cameras == null || widget.cameras.length < 1) {
            print('No Cameras Found.');
        } else {
            controller = new CameraController(
                widget.cameras[0],
                ResolutionPreset.high,
            );
            controller.initialize().then((_) {
                if (!mounted) {
                    return;
                }
            });
        }
    }

    void start() {
        controller.startImageStream((List<dynamic> list) {
            setState(() {
                isDetecting = true;
            });
            widget.setRecognitions(list);
        });
    }

    void stop() {
        controller.stopImageStream();
        controller.dispose();
    }
}

```

```

    }

    setState(() {});

    controller.startImageStream((Cameralmage img) {
        if (!isDetecting) {
            isDetecting = true;
            Tflite.detectObjectOnFrame(
                bytesList: img.planes.map((plane) {return plane.bytes;}).toList(),
                model: "SSDMobileNet",
                imageHeight: img.height,
                imageWidth: img.width,
                imageMean: 127.5,
                imageStd: 127.5,
                numResultsPerClass: 1,
                threshold: 0.4,
            ).then((recognitions) {
                /*
                    When setRecognitions is called here, the parameters are being passed on to the parent
                    widget as callback. i.e. to the LiveFeed class
                */
                widget.setRecognitions(recognitions, img.height, img.width);
                isDetecting = false;
            });
        }
    });
});

}

@Override
void dispose() {
    controller?.dispose();
    super.dispose();
}

@Override
Widget build(BuildContext context) {
    if (controller == null || !controller.value.isInitialized) {
        return Container();
    }

    var tmp = MediaQuery.of(context).size;
    var screenH = math.max(tmp.height, tmp.width);
    var screenW = math.min(tmp.height, tmp.width);
    tmp = controller.value.previewSize;
    var previewH = math.max(tmp.height, tmp.width);
    var previewW = math.min(tmp.height, tmp.width);
    var screenRatio = screenH / screenW;
    var previewRatio = previewH / previewW;

    return OverflowBox(

```

```

maxHeight:
    screenRatio > previewRatio ? screenH : screenW / previewW * previewH,
maxWidth:
    screenRatio > previewRatio ? screenH / previewH * previewW : screenW,
child: CameraPreview(controller),
);
}
}

import 'package:flutter/material.dart';
import 'dart:math' as math;

class BoundingBox extends StatelessWidget {
final List<dynamic> results;
final int previewH;
final int previewW;
final double screenH;
final double screenW;

BoundingBox(
this.results,
this.previewH,
this.previewW,
this.screenH,
this.screenW,
);

@Override
Widget build(BuildContext context) {
List<Widget> _renderBox() {
return results.map((re) {
var _x = re["rect"]["x"];
var _w = re["rect"]["w"];
var _y = re["rect"]["y"];
var _h = re["rect"]["h"];
var scaleW, scaleH, x, y, w, h;

if (screenH / screenW > previewH / previewW) {
scaleW = screenH / previewH * previewW;
scaleH = screenH;
var difW = (scaleW - screenW) / scaleW;
x = (_x - difW / 2) * scaleW;
w = _w * scaleW;
if (_x < difW / 2) w -= (difW / 2 - _x) * scaleW;
y = _y * scaleH;
h = _h * scaleH;
} else {
scaleH = screenW / previewW * previewH;
scaleW = screenW;
var difH = (scaleH - screenH) / scaleH;
}
}
}
}

```

```

x = _x * scaleW;
w = _w * scaleW;
y = (_y - difH / 2) * scaleH;
h = _h * scaleH;
if (_y < difH / 2) h -= (difH / 2 - _y) * scaleH;
}

return Positioned(
  left: math.max(0, x),
  top: math.max(0, y),
  width: w,
  height: h,
  child: Container(
    padding: EdgeInsets.only(top: 5.0, left: 5.0),
    decoration: BoxDecoration(
      border: Border.all(
        color: Color.fromRGBO(37, 213, 253, 1.0),
        width: 3.0,
      ),
    ),
    child: Text(
      "${re["detectedClass"]}\ ${(re["confidenceInClass"] * 100).toStringAsFixed(0)}%",
      style: TextStyle(
        color: Color.fromRGBO(37, 213, 253, 1.0),
        fontSize: 14.0,
        fontWeight: FontWeight.bold,
      ),
    ),
  ),
);
);

},
);
}
).toList();
}

return Stack(
  children: _renderBox(),
);
}
}

```

Static Camera Object Detection

```
import 'dart:io';
import 'package:flutter/material.dart';
import 'package:swipedetector/swipedetector.dart';
import 'package:image_picker/image_picker.dart';
import 'package:tflite/tflite.dart';

class StaticImage extends StatefulWidget {

  @override
```

```

    _StaticImageState createState() => _StaticImageState();
}

class _StaticImageState extends State<StaticImage> {
  File _image;
  List _recognitions;
  bool _busy;
  double _imageWidth, _imageHeight;

  final picker = ImagePicker();

  // this function loads the model
  loadTfModel() async {
    await Tflite.loadModel(
      model: "assets/models/ssd_mobilenet.tflite",
      labels: "assets/models/labels.txt",
    );
  }

  // this function detects the objects on the image
  detectObject(File image) async {
    var recognitions = await Tflite.detectObjectOnImage(
      path: image.path,    // required
      model: "SSDMobileNet",
      imageMean: 127.5,
      imageStd: 127.5,
      threshold: 0.4,     // defaults to 0.1
      numResultsPerClass: 10, // defaults to 5
      asynch: true        // defaults to true
    );
    FileImage(image)
      .resolve(ImageConfiguration())
      .addListener((ImageStreamListener((ImageInfo info, bool _)) {
        setState(() {
          _imageWidth = info.image.width.toDouble();
          _imageHeight = info.image.height.toDouble();
        });
      }));
    setState(() {
      _recognitions = recognitions;
    });
  }

  @override
  void initState() {
    super.initState();
    _busy = true;
    loadTfModel().then((val) {{
      setState(() {
        _busy = false;
      });
    });
  }
}

```

```

    });
}

// display the bounding boxes over the detected objects
List<Widget> renderBoxes(Size screen) {
if (_recognitions == null) return [];
if (_imageWidth == null || _imageHeight == null) return [];

double factorX = screen.width;
double factorY = _imageHeight / _imageHeight * screen.width;

Color blue = Colors.blue;

return _recognitions.map((re) {
return Container(
child: Positioned(
left: re["rect"]["x"] * factorX,
top: re["rect"]["y"] * factorY,
width: re["rect"]["w"] * factorX,
height: re["rect"]["h"] * factorY,
child: ((re["confidenceInClass"] > 0.50)) ? Container(
decoration: BoxDecoration(
border: Border.all(
color: blue,
width: 3,
),
),
child: Text(
"${re["detectedClass"]} ${(re["confidenceInClass"] * 100).toStringAsFixed(0)}%",
style: TextStyle(
background: Paint()..color = blue,
color: Colors.white,
fontSize: 15,
),
),
),
),
),
),
);
}),
);
}).toList();
}

@Override
Widget build(BuildContext context) {
Size size = MediaQuery.of(context).size;

List<Widget> stackChildren = [];

stackChildren.add(
Positioned(
// using ternary operator
child: _image == null ?
Container(

```

```

child: Column(
  mainAxisAlignment: MainAxisAlignment.center,
  children: <Widget>[
    Text(
      "\n\nSelect an Image",
      style: TextStyle(color: Colors.white, fontSize: 25)
    ),
    Expanded(
      child: SwipeDetector(
        onSwipeUp: () {
          print("Swiped Up");
        },
        onSwipeDown: () {
          print("Swiped Down");
          Navigator.pop(context);
        },
        onSwipeLeft: () {
          print("Swiped Left");
          // ignore: unnecessary_statements
          getImageFromCamera();
        },
        onSwipeRight: () {
          print("Swiped Right");
          // ignore: unnecessary_statements
          getImageFromGallery();
        },
      )),
    // SizedBox(
    //   width: 200.0,
    //   height: 600.0,
    //   child: Center(
    //     child: Text(
    //       "Select an Image",
    //       maxLines: 1,
    //       style: TextStyle(color: Colors.white, fontSize: 25)
    //     ),
    //   )),
  ],
),
)
: // if not null then
Container(
  child:Image.file(_image)
),
);
);

stackChildren.addAll(renderBoxes(size));

if (_busy) {
  stackChildren.add(

```

```
        Center(
            child: CircularProgressIndicator(),
        )
    );
}

return Scaffold(
    appBar: AppBar(
        title: Text("Object Detector"),
        centerTitle: true,
    ),
    floatingActionButton: Row(
        mainAxisAlignment: MainAxisAlignment.end,
        children: <Widget>[
            FloatingActionButton(
                heroTag: "Fltbtn2",
                child: Icon(Icons.camera_alt),
                onPressed: getImageFromCamera,
            ),
            SizedBox(width: 10,),
            FloatingActionButton(
                heroTag: "Fltbtn1",
                child: Icon(Icons.photo),
                onPressed: getImageFromGallery,
            ),
        ],
    ),
    body: Container(
        alignment: Alignment.center,
        child: Stack(
            children: stackChildren,
        ),
    ),
);
}
// gets image from camera and runs detectObject
Future getImageFromCamera() async {
    final pickedFile = await picker.getImage(source: ImageSource.camera);

    setState(() {
        if(pickedFile != null) {
            _image = File(pickedFile.path);
        } else {
            print("No image Selected");
        }
    });
    detectObject(_image);
}
// gets image from gallery and runs detectObject
Future getImageFromGallery() async {
    final pickedFile = await picker.getImage(source: ImageSource.gallery);
```

```
setState(() {
  if(pickedFile != null) {
    _image = File(pickedFile.path);
  } else {
    print("No image Selected");
  }
});
detectObject(_image);
}
```