TALLINN UNIVERSITY OF TECHNOLOGY
SCHOOL OF ENGINEERING
Department of Electrical Power Engineering and Mechatronics

CALCULATING THE VOLUME OF BULK MATERIALS USING MODERN PORTABLE DEVICES

PUISTEMATERJALI MAHU ARVUTAMINE KAASAEGSETE KAASASKANTAVATE SEADMETE ABIL

MASTER THESIS

Student: ZAHEER IQBAL

Student code: MAHM 195990

Supervisor: Researcher, Dr. Dmitry Shvarts

Tallinn 2021

(*On the reverse side of title page*)

AUTHOR'S DECLARATION

Hereby I declare, that I have written this thesis independently.
No academic degree has been applied for based on this material. All works, major viewpoints and data of the other authors used in this thesis have been referenced.

"....." ..... 20…..

Author: .....
/signature /

Thesis is in accordance with terms and requirements

"....." ..... 20….

Supervisor: ….....
/signature/

Accepted for defence

"....."..... 20… .

Chairman of theses defence commission: .....
/name and signature/

Non-exclusive License for Publication and Reproduction of GraduationTthesis[1]

I, Zaheer Iqbal (name of the author) (date of birth: 5th October 1981) hereby

1. grant Tallinn University of Technology (TalTech) a non-exclusive license for my thesis

"Calculating the volume of the bulk materials using modern portable devices"

"Puistematerjali mahu arvutamine kaasaegsete kaasaskantavate seadmete abil",

(*title of the graduation thesis*)

Supervised by

Researcher, Dr. Dmitry Shvarts,

(*Supervisor's name*)

1.1 reproduced for the purposes of preservation and electronic publication, incl. to be entered in the digital collection of TalTech library until expiry of the term of copyright;

1.2 published via the web of TalTech, incl. to be entered in the digital collection of TalTech library until expiry of the term of copyright.

1.3 I am aware that the author also retains the rights specified in clause 1 of this license.

2. I confirm that granting the non-exclusive license does not infringe third persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

---

[1] *Non-exclusive Licence for Publication and Reproduction of Graduation Thesis is not valid during the validity period of restriction on access, except the university`s right to reproduce the thesis only for preservation purposes.*

_____ (*signature*)

_____ (*date*)

Department of Electrical Power Engineering and Mechatronics

THESIS TASK

Student: **Zaheer Iqbal**

Study programme: MSc

main speciality: Mechatronics

Supervisor(s): Researcher, Dr. Dmitry Shvarts

Consultants: ……..(name, position)

…… (company, phone, e-mail)

Thesis topic:

(in English) **Calculating the volume of the bulk materials using modern portable devices**

(in Estonian) **Puistematerjali mahu arvutamine kaasaegsete kaasaskantavate seadmete abil**

Thesis main objectives:

1. To propose a solution for bulk volume calculation using cell phone
2. Experimentation and Evaluation of different Photogrammetric techniques
3. Development of an android application for calculation of bulk volumes.

Thesis tasks and time schedule:

No	Task description	Deadline
1.	Literature review	30th November 2020
2.	Introduction Chapter	20th December 2020
3.	Experimental results and Optimization	28th February 2021
4.	Application Development	30th April 2021
5.	Thesis Defense	1st June 2021

Language: …… **Deadline for submission of thesis:** "....."..... 20….a

Student: …….. ..... "....."…….....20….a

/signature/

Supervisor: …… …….. "....."..... 20….a

/signature/

Consultant: …… …..... "....."..... 20….a

/signature/

Head of study programme: …… ..... "....."..... 20…..a

/signature/

Terms of thesis closed defence and/or restricted access conditions to be formulated on the reverse side

TABLE OF CONTENTS

PREFACE

The thesis topic is very appealing for enthusiasts in computer vision and image processing. The main attraction of the thesis topic is its endless room for further exploration and numerous current applications in digital transformation. The thesis work comprises over process involved in developing Android Application for calculating the volume of bulk materials through image processing. The work includes number of approaches and techniques used to build a system compatible for portable devices keeping the computational budget and accuracy in view. Each step of the whole development process is described in details and can be productive for readers interested in Android development and computer vision.

Dr Dmitry Shvarts and Prof. Mart Tamre has performed a remarkable work on the subject, in 2014. This work remained the prime inspiration for the researcher for further exploration on the subject.

I would like to express heartfelt gratitude to Professor Mart Tamre and Department of Electrical Power Engineering and Mechatronics for accepting me in Master Studies and providing me such a significant study experience.

I would like to specially thank my father for all his support in my educational endeavors. I would like to thank Dr Dmitry Shvarts under whose supervision this thesis work is completed, whose constant availability and determined guidance has made it possible to accomplish all the challenging tasks.

LIST OF ABBREVIATIONS

3D-Three Dimensional
GmbH-Gesellschaft mit beschränkter Haftung
CIE-International Commission on Illumination
ROI-Region of Interest
CRP-Close-range photogrammetry
SfM- Structure from motion
PMVS-Patched Based Multi View Stereo
UAV-Unmanned Air vehicle
DTM-Demographic Transition Model
DEM-Digital Elevation Mode
RTK-Real Time Kinematic
RGB-Red Green Blue
POI-Poin of Interest
SLV-Structured Light Vision
CCS-Camera Coordinate System
CNN-Convolution Neural Network
FCNN-Fourier Convolution Neural Network
RGBD-Red Green Blue Depth (Imaging)
SLAM-Simultaneous Localization and Mapping
USDA-United States Department of Agriculture
TIN-Triangulated Irregular Network
RMSE-Root Mean Square Error
GCP-Ground Control Points
CP-Check Point
Lidar-Light Detection and Ranging
GPS-Global Positioning System
INS-Inertial Navigation System
RANSAC-RANdom SAmple Consensus
SDK-Software Development Kit
MUSEFood- Multi sensor based food volume estimation
FCN-Fully convolutional networks
MLS-Maximum Length Sequence
SIFT-Scale Invariant Feature Transform
PSR-Poisson Surface Reconstruction
SGM-Semi Global Matching

API-Application Programming Interface

App-Aplication

IDE-Integrated Development Environment

HSV Hue Saturation and Value

ML-Machine Learning

APK-Android Package

1. INTRODUCTION

1.1. Overview

Volume measurement of bulks i.e pile of timber, heap of pipe lengths, and other similar commodities is of crucial importance for several industries which include storage, logistics, transportations and construction companies and many other similar businesses. An accurate, convenient and inexpensive estimation of such irregular shaped bulk items can lead to more efficient shipping and avoid inefficient costing at several stages throughout whole shipment track. There are numerous other applications in businesses like mining, construction, timber, surveying etc., which require bulk volume calculation on every step. Incorrect volume estimation can lead to under loading and can require additional transportation movements to shift the same amount of material, hence reducing profitability.

There are several solutions available in the market for calculation of bulk material volumes but the accurate ones are with a higher cost and require some dedicated hardware i.e special cameras, laser projectors, UAV etc. Computational needs for these solutions are also high and together with special equipment, it sources inconvenient use.[1]

In this thesis, a solution is proposed based on a close-range photogrammetry by portable devices i.e mobile phone camera, for the calculation of the log pile volume. The system segments the material of interest in the scene. An android application is developed using modern CameraX API to capture the image of log piles. Object segmentation approaches are discussed in detail to be integrated with the application to localize the region of interest. The extracted regions were then processed with OpenCV for android to get the on-screen area and subsequently the volume. Onscreen pixel based volume is then converted into matric units to be displayed on application screen.

The system workflow consists following major steps,

- Image Acquisition
- ROI Segmentation
- Feature Extraction
- Perspective Transformation of pixel space to physical space (Matric Units)
- Volumetric Calculation

It is a difficult task to make an image suitable for feature extraction. Before getting features, various image pre-processing techniques like resizing, normalization, brightness adjustments etc. are applied on the acquired image. The image size and optimization need to be kept for mobile GPUs to reduce the processing time and make computationally intensive computer vision algorithms feasible for a mobile device. Success of the result is measured with the degree of accuracy of volumes calculated with the proposed system. As a model case a pile of timber is practiced for volume calculations and the result are compared with conventional manual measurement procedure.

1.2. Motivation

It will not be untrue if we term Image processing as the most prominent domain which comes under computer vision discipline. The importance of computer vision can easily be estimated with keeping in mind the problems it solves. Image processing through computer vision makes the bridge between real world and digital world. We are in an era where digital transformation is running at an unprecedented pace. Image processing plays an important role in almost all AI applications from attendance systems to self-driving vehicles. Smart phone technologies are also developing in a fast pace to support us in numerous fields ranging from children education to health monitoring systems for elderly people.

The author is passionate about exciting, contemporary and fast developing fields of knowledge. The topic for thesis work was chosen for its wide span of coverage from image processing to android application development. The motive behind selection of the topic was to discover new techniques and technologies which can be utilized in a wider extent in course of educational and professional endeavors.

1.3. Objective

The main objective of this thesis work is to propose a solution to calculate bulk volume measurement with help of mobile device. To develop an android App capable of extracting the desired object/objects from an image with image processing and machine

learning techniques. Analyze the image and extract the required information, perform calculations and present the result in form of calculated volume. Following course of action is followed to achieve the objectives of the thesis work.

- Review of literatures and available solutions for bulk volume measurement on mobile devices
- Review of all the possible approaches.
- Finalizing the approach on basis of performance indicators
- Development of the resources and software tools
- Development of Android App
- Testing and evaluation

1.4. Thesis Structure

Description of contents of all the sections in this thesis are summarized as bellow to give a preview to the reader of this document.

Chapter 1 covers Overview of the this work along with motivation behind this work, Objectives and thesis structure are also described in this chapter.

Chapter 2 contains the review of researches related to the topic of this thesis. A brief description of the selected solutions on the addressed problem is also included in this chapter. In the end of this chapter all reviews are summarized.

Chapter 3 is about all the techniques and methods used in this work. It covers different stages of the thesis work and approaches considered for developing the solution.

Chapter 4 comprises the details on development of the solution. All the steps for creating an optimum solution are described in details.

Chapter 5 provides the discussion on the complete project achievements, limitations, Future work suggestions and directions. This chapter also includes thesis summary both in English and Estonian Languages.

2. LITERATURE REVIEW

This Chapter describes the existing solutions, problems and research performed on the measurement of bulk materials through photogrammetry techniques.

2.1. Close Range Photography

A remarkable work on the subject was performed by Researcher Dmitry Shvarts and Prof. Mart Tamre, 'BULK MATERIAL VOLUME ESTIMATION METHOD AND SYSTEM FOR LOGISTIC APPLICATIONS", 9th International DAAAM Baltic Conference "INDUSTRIAL ENGINEERING" April 2014, Tallinn, Estonia.[2] This publication remains the prime inspiration for the researcher for further exploration on the subject.

In this research, Bulk material volume assessment if performed with machine vision technology and smart algorithms on a mobile device. The study was carried out in 2014 and the method was discussed on the example of the timber volume estimation.

Color is considered as the basic key for segmentation but due to outdoor application, the colors were not homogenous, the developers came with a new solution to the problem and used Chromaticity instead. Chromaticity is an objective description of the quality of a color regardless of its brightness. The study proposes feature extraction considering the chromaticity in term of two characteristics: hue and saturation. The two commonly known are HSV (Hue Saturation Value/Brightness) and CIE LCh (CIE stands for "International Commission on Illumination", and Lxcxh means L – Lightness, c – Chroma or "saturation", h- Hue).

Range of hue (H) and saturation values (S) that correspond to the timber are used to separate the object i.e timber from the background. The resulting logical image still had noises. Many pixels outside the timber were classified as a timber due to holes, dirt and shadows from the nearby logs. Researchers eliminated the remaining noise by morphological opening and closing operation with a small kernel.

The developed algorithm was tested on the phone SONY XPERIA GO with a 5Megapixel camera and 512 MB of RAM. The image was taken by standard Android application. The main window of the application asks the user to enter the timber length manually and then to load the timber image. Some object of the known size in the scene is used for calibration at the same distance from the camera as the ROI.

Another study uses digital close range photogrammetric method for volume determination. [3] The test was conducted on a physical model and actual object. The result of the calculated volume from close range photogrammetry data was compared with the conventional method to examine the accuracy of computation. It is found that better land surface representation can give good accuracy of volume generation which depends on number of elements including Three dimensional coordinates of the target point, point distribution and interpolation methods. The total volume of target points is the measure of whole surface with respect the reference surface. The preferred accuracy can be achieved when the objects are measured accurately from classical photographs or from digital images captured from a short distance.

Close-range photogrammetry (CRP) applications mostly have a working distance of less than 300 m between object and camera. It is the reconstruction of object at the same time from several images and the images are captured from different viewpoints. The best possible viewpoint is used to obtain suitable geometry of intersecting rays and strong network. The determination of the 3D coordinates from a definite point is achieved through the intersection of two or more straight lines in bundle adjustment process. 3D coordinate determination through photogrammetry is based on the collinearity equation, which states that object point, camera projective center and image point lie on a straight line.

2.2. Volumetric APP

APD Volumetric App, a stand-alone native Android application enables the smartphone to calculate the volume of the substance in a container through image analysis. [4] Using pixel ratios, a much more accurate estimation of the remaining liquid can be made. This allows the rapid quantification of the contents without the need of volumetric cylinders, especially when dealing with very minute quantities. With this app, more accurate estimations to keep track of reagents for logistics, leisure, and operations, can be performed in an easily streamlined and convenient manner.

The research published in 2017 purposes for measuring the volume of reagents in laboratories. The application was developed using the Eclipse IDE, it estimates the remaining reagent volume after taking a close image of the substance in the container, A user-defined cropped image is used for the auto-calculation of content space from the known maximum volume of the container. The volume calculation is achieved by

calculating the pixels to volume ratio as the volume of the whole container is known. This gives volume of the single pixel and by multiplying with the pixels of the substance.

The volume calculations can be determined by pixel and highly depends on the outlining/cropping process the researchers proposed the use of smartphone pen with the soft round ball at the end, or Specialized pens that are present in Microsoft Surface Pro series, Samsung Tablets, and Samsung Galaxy Note series to outline the images in order to maintain the accuracy of the measurement. Photos taken from above or below liquid meniscus level can cause parallax errors that will produce inaccurate estimates. This application also has the very specific use with higher user intervention in calculating the quantity/volume. It works only with the content of round-bottomed or cylindrical objects such as beakers, test tubes, and microfuge tubes as these are the typical shapes in biomedical research labs.

There are several mobile phone applications which already exist with the tag name of calculating volume e.g "Calculator bulk materials", "Volume Calculator" by swartland Apps, these applications give volume of regular shapes only with maximum user intervention. "Measurement Master" by Robert Bosch Power tools Gmbh, is another example picked on the basis of ranking, the application is specifically designed for civil engineering perspective and interior designing, most of functions require additional hardware i.e laser range finder, thermal camera etc. "Surface and Volume Measurement" by Web Dream is also a popular solution but the applicability is limited and accuracy is not guaranteed.

2.3. SfM-Patched Based Multi View Stereo System

Research carried out by Zhang Chunsen and Zhang Qiyuanpaper proposed volumetric calculation [1] for large heap of coal by using DJI M600 six-rotor UAV equipped with a Sony ILCE6000 camera.

The 24.3 MP digital camera obtains the target high resolution, large overlapped images taken from an angle and then these images are processed based on the SfM-Patched Based Multi View Stereo Vision. Feature extraction is done through feature matching and point cloud matching based on PMVS. Then, image control points are introduced into bundle adjustment for obtaining the precise coordinates in the ground coordinate system. Finally, the DTM method is used to calculate the volume of the heap after doing point cloud segmentation.

This paper combines the technical compensations of UAV and SfM-PMVS to study the use of drone-equipped digital cameras to collect tilt images of the TX01 coal pile in Tongchuan (China). Based on the collected images, a dense 3D point cloud of the coal pile was acquired using a motion recovery structure and a multi-view stereoscopic vision method. The ground image point was adjusted to obtain the accuracy of the point cloud in the ground measurement.

A method proposed in another research to estimate volume of large wood accumulation using Structure from Motion (SfM) photogrammetry through UAV.[5] The method aims to develop accurate 3D models of prototype LW accumulations. To endorse optimum overlap for high quality digital DEM reconstruction, the acquired datasets comprised 72 and 120 images taken from an angle.

Images were taken from positions 2 meters above ground. Large number of images acquired to get as much as possible details through 360 degrees movement of UAV around the wood accumulates. Commercially available SfM software package, Pix4DMapper was used for processing including point matching and point cloud generation and adjusting internal and external camera parameters.

Large Wood accumulation was separated from the rest of the point cloud model using manual segmentation tools in CloudCompare. Explicit triangulation methods and modelling of the point cloud is used for surface reconstruction.

The researchers used three meshing approaches, a simplified mesh model based on Delaunay triangulation, Delaunay triangulation with Voronoi filtering in CloudCompare, PSR Poisson Surface Reconstruction in MeshLab. All meshes were filtered for extracting isolated pieces in MeshLab, for appropriate volumetric computation of the generated 3D Large Wood accumulation models. Volumes for the 3D point cloud and mesh models were computed through two software workflows: a predefined Pix4D tool ‘Volumes’, and CloudCompare’s ‘Compute 2.5D Volume’ tool. The method is a detailed work and used for accurate volume measurements and 3D modelling along with that for higher number of datasets the computational needs are very huge.

2.4. Orthogonal Imaging

In this research paper published in 2017, a new method for volume estimation is proposed. The technique estimates the volume of food products from three orthogonal images without explicit 3D reconstruction of the shape of product. [6] The proposed approach makes no detailed assumption about the shape of the object, except that it is convex, axially symmetric, and axially aligned.

The study was conducted on eight different fruits and vegetables and eight artificial objects with known volume (sphere, cube, cylinder, etc). Three-image acquisition systems were used to capture the images, the objects were carefully positioned to follow the assumptions of axial alignment. PointGrey and Flea2 RGB color cameras with 8 mm lens were used in the experiments. Camera angles were set to be orthogonal and there was a uniform back ground behind the object in order to extract the POI.

To convert the outline information from image coordinates in pixels to real-world coordinates in units of length, one of the three cameras was selected as reference, (e.g., the camera providing the x - z projection). Distance of camera to the center of the object is then measured, images from the other cameras are also rescaled for calculation to simulate the setup that all cameras are placed at equal distances. The method has of a very specific scope, Food objects studied is limited to spherical and ellipsoidal shapes photographed under ideally arrange conditions.

2.5. 3D Modelling and Reconstruction

The study published in 2017 introduces method consists in photographic recording of one or two ends of the log pile following by image processing with uniquely designed software to calculate the volume of log pile. [7] The software performs the automatic detection of visible borders of each image, following recovery of the log pile 3-D structure and measurement of its volume in the end are performed during the processing.

Two coordinate systems are introduced to describe the geometric model of the camera: pixel system (Xi,Yi) according to the photogrammetry theory, Hence the digital image is processed the measuring unit is pixel and spatial coordinate system (X,Y,Z). Transit

between coordinate systems which determines the connection of pixel and three-dimensional

$$X_i = \left(\frac{f}{Z}\right).X \tag{2.1}$$

$$Y_i = \left(\frac{f}{Z}\right).Y \tag{2.2}$$

X,Y,Z – spatial dimension and distance to the object, Xi,Yi – pixel dimension of the object image and f – focal distance of the camera lens. A known volume object is used for calibration and is required to be located in the abuts plane and visible for detecting by the software. Reconstruction of the 3D coordinates of the logs is achieved by one or two images of the log pile ends. It is necessary to determine the one-to-one correspondence of the abut from the one end of the pile to the abut from the other so the both ends of the log would be assigned. The researcher here applies the Information about orientation and stem taper for achieving maximum accuracy, In case of one image the remaining parameters are assumed for the other end of the pile. The 3D model rendering is implemented with DirectX API. The method gives relatively good measurements fulfilling the required industrial standards.

Another research paper discusses the generation of a Three Dimensional model by taking views of the object from different angles. [8] Image acquisition is done using a single digital camera and turn table. Camera calibration is done using the images of checkerboard pattern. Camera calibration gives the intrinsic and extrinsic parameters of a camera. These camera parameters are used to calculate the camera matrix. Camera matrix gives the relation between image reference frame and world reference frame. Each view captured of the object is segmented. The silhouettes are obtained from the segmented images of object. Visual hull is formed by the silhouettes and camera projection matrix information using space carving algorithm. Finally, the 3D Model of the object is constructed. The camera used is a Kodak-PixPro, FZ51 60-Megapixels braced on a tripod. The camera is so adjusted that the maximum view of object can be seen in the image, the object is placed in the center of a rotatable platform. The camera parameters can be used to calculate projection matrix. Projection matrix is the link between world and image coordinates. The camera parameters are used as functions of entries of the projection matrix. 3D modelling methodology is applied using sequence of images of the object rotating about single axis. Fundamental matrix computation is used for deriving camera projection matrix. The Space Carving Algorithm for photometric reconstruction is used.

Carving is done on the initial bounding volume by all the camera views. The voxel-based 3D modelling gives volume in the form of number of voxels remaining after carving. To convert the volume in matric unit like, multiply the residual voxel count by three-dimensional calibration factor. This factor represents the equivalence between the number of voxels and one centimeter. The proposed method is able to calculate the volume just 1% to 2% error. The application of the method is limited to geometrical shaped objects with a very close range and specially arranged hardware.

A non-contact volume measurement method for irregular objects is proposed in a research paper based on 3D reconstruction technology, with use of linear laser camera.[9] The line-surface model of laser surface and feature points are first established in a camera coordinate system. The point cloud coordinates of measured object are calculated using the geometric measurement principle and transformed into the world coordinate system from the camera coordinate system. The differential object point cloud is obtained by subtracting the reconstructed model from the initial background model, and the object volume is calculated by integrating the differential depth point cloud data. To extract accurate center of light stripe, truncated Gaussian distribution is used as a fitting algorithm. The measurement error was less than 4.5% at the distance of 2 meters. The experimental results show that the proposed method can accurately reconstruct 3D objects and accurately measure the irregular objects volume.

The laser projects a light beam to the object, and the stripe of light is deformed by the changes of depth of object surface. Then the distorted light stripe is captured by camera and measured to the depth (h) of a light stripe. The pattern measures the information of a light stripe at a time and it can get three-dimensional information of object because of the movement of object. The line-surface model calculated by optical plane calibration is used to establish the relative position of the camera and the laser. After triangulation, the method fitting the light stripe by truncated normal distribution is used. The collected stripe images of object are pre-processed and the center coordinates of stripe are extracted. The calibration parameters are used to convert the center coordinates of stripe to the world coordinate system, and the real three-dimensional coordinates of the object are obtained to calculate the volume.

2.6. Structured Light Vision

The proposed system consists of a novel 2x2 laser line grid projector, a sensor, and software modules.[10] Laser-modulated images of boxes are used for volume measurement. For laser-modulated images, deep learning model is proposed by using a holistically nested edge detection network to extract edges. A calibration method for the line-structured light projector is used. The method uses information that is extracted from the structure edges of the measured boxes, which can only be computed when at least two of their faces are projected by the laser projector. The detailed workflow is listed as

- System parameters are obtained first by using calibration method
- The visual sensor connected to a portable device is used. Two images of any two adjacent faces of the box are achieved. The four modulated laser bands should intersect the four edges of the box face.
- The system will process the collected images and then obtain the box length, width, and height. Finally, the system calculates the volume of the measured box.

The projection from a 3D point P(Xw, Yw, Zw) in the WCS to a 2D image point p(u, v) in the image plane is expressed by the following equation [10]

$$\rho \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & \delta & u_\circ & 0 \\ 0 & \beta & v_\circ & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ O^T & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} \tag{2.3}$$

$$A = \begin{bmatrix} \alpha & \delta & u_\circ \\ 0 & \beta & v_\circ \\ 0 & 0 & 1 \end{bmatrix}', \qquad R = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix}, \qquad T = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix}$$

Where T and R represent the translation vector and rotation matrix from the coordinate system to the CCS, respectively. α and β are the scale factors in u and v axes of the camera, respectively, and δ is the skew of the two image axes. ρ is a nonzero factor, and ($u_\circ$, $v_\circ$) is the principal point. The rotation matrix R and translation vector T, which translate to a 3D point Pc (xc, yc, zc) in the CCS, encapsulate the camera orientation and position. The method employs FCNN which has addressed the problem of detecting edge and object boundaries in natural images. The method successfully provides volume of boxes with accuracy but the applicability is limited to the defined shaped boxes with utilization of specialized hardware for laser projector.

Another paper proposes a dynamic volume measurement system based on the line structured light vision sensor.[11] Light strip center is extracted with the Steger algorithm and then method of Zhengyou Zhang calibration is used to obtain camera intrinsic parameters. Light plane equation in the camera coordinate system is obtained by calibrating the light plane by rotation matrix, translation matrix and homographic matrix obtained from internal parameters. By solving the simultaneous equations three dimensional coordinates are calculated and 3D point clouds are obtained. By integration of the 3D point cloud information, volume of the object is calculated.

In the proposed method, light stripes coordinate information in the image and the light plane equation is used to reconstruct 3D model to calculate the volume. The system consists of a high-definition camera, a processing platform, a transmitting station and a laser projector. Realtime three-dimensional coordinates of the light spot are constructed with the captured image. Distance between the obtained point and the material is calculated with the horizontal integration, and the volume can also be obtained with the vertical integration. Laser stripes in the image in the width direction are symmetrical and occupy more than one pixel similar to the Gauss distribution. Steger method is used to extract the center of light stripe. To reduce the number of computations, only the effective area of the interception of the light stripe center is extracted.

The 3D coordinates of the light stripe in the measurement range are obtained by using the reconstruction method without the measured object on the measurement platform. Then the object is placed on the platform which deforms the light stripes then again, the 3D coordinated are obtained and the distance is calculated by the distance formula. Gray point industrial camera, a 650nm laser, a 100mW red word laser projector, and a 25mm board grid calibration target is used for the method.

2.7. RGB-D Camera

This paper presents a potato volume measurement method based on RGB-D camera. [12] An RGB-D camera established based on binocular stereo vision was used in this method. The method was developed to measure volume of potatoes and built utilizing an image processing algorithm for depth images. The experiment was conducted with the depth image collection and volume detection of 120 potatoes.

The sampled potatoes were divided into regular shaped and irregular shape group. As per the results of the method the prediction error of normal potatoes was 9%; the prediction error of irregular potatoes was 30%. A depth camera supports a ranging function and has the ability to measure the features in the 3-dimensional space as compared with conventional RGB Camera. The range image generated by the depth camera can display the distances to points in a scene with reference to a specific point, and also shows the pixel values corresponding to the distances. These distances can be converted to matric units with employing adequate calibration techniques with better accuracy levels. The depth camera also has the functionality to record the distance of object from the camera lens in a 16 bit integer matrix format.

Binocular Stereos Vision sees one scene from two different viewpoints. With the help of principle of triangulation, the information of depth is calculated by the disparities. A known volume cube is used to calibrate the camera. The system calculated distance from camera for different points on the potato surface with binocular vision. Then grey scale image is used to calculate grey value of each point in order to calculate height of each point on the potato surface. To measure the volume of potato, an algorithm is used to integrate the top-bottom volume acquired from potato depth images.

2.8. Simultaneous Localization and Mapping (SLAM)

Another method utilizes specialized simultaneous localization and mapping (SLAM), a modified version of convex hull algorithm, and a 3D mesh object reconstruction technique.[13] This paper explores the feasibility of applying SLAM techniques for continuous food volume measurement with a monocular wearable camera. Food detection and segmentation part was performed using convolution neural network (CNN). Once the food volume is measured by the wearable device, the data can be fused with USDA national nutrient database for further dietary analysis. The visual SLAM framework can be divided into four parts including visual odometry, loop closure, back-end optimization and mapping. Visual odometry aims to estimate the camera's position by analyzing the feature points on the captured image between different views, in order to compute the trajectory of the moving camera.

Loop closure is the process of recognizing the location which has been previously captured in order to correct the drift trajectory of the camera. Back-end optimization processes information from visual odometry and loop closure, Once obtained the optimized camera's trajectory, mapping is used to build an environment map through

captured image sequence. Convex hull algorithm is used for 3D reconstruction which fits best with SLAM. Apple iPhone 6 plus and a 4k wearable action camera have both been used for data collection to explore the possibility of continuous food volume measurement.

In the process of capturing video data, target object has been placed on a black background. A Rubik cube has been designed as a scale reference for calibration. With the use of the statistical outlier filter, the point completion technique and the multiple convex hull algorithm, the proposed technique can get a performance with an overall accuracy of 83%.

2.9. Other Approaches

Another study focusses on estimating the volume of stockpile using UAV [14] and comparing with terrestrial laser scanner for planning purpose, data collection, data processing and calculating volume of stockpile. From the obtained data, height of surface points of stockpile volume is interpolated to form a triangle known as Triangulated Irregular Network (TIN).

A UAV method was used to collect aerial photo data after a proper flight planning and the terrestrial laser scanner was used to collect the data from the ground. Ground control points (GCPs) and checkpoints (CPs) established to obtain a tie point with georeferenced and improve the relative orientation. The study produced DEM digital elevation model and contour for the volume measurements. The quality of the dense cloud high parameters been applied and moderate depth filtering. The volume extract from the UAV is compared to data from laser scanner to determine the accuracy and capability of the data and instrument.

Another research paper presents a volume measurement approach of the sand carrier using unmanned aerial vehicle (UAV) imagery.[15] The fine detailed surfaces of the sand carriers are reconstructed from dense point clouds derived by UAV-based mapping. Then, the volume of sand is calculated by the differential method, which multiplies the height difference between the UAV-derived 3D surfaces of the vessel and sand by the resolution of these surfaces. The study was carried out on 10 sand carriers based on the result the absolute values of the relative deviation between the calculated volume and the reference volume was approximately 2%.

A Study for volume calculation of collected lunar soil proposed method is designed to fully use the sensors already installed as there was no additional installation position for volume measurement equipment. [16] The designed method is based only on a single camera. It uses a sequence of images of the collection area captured by the camera mounted on the acquisition arm to accurately reconstruct the terrain of the collection area surface before and after soil acquisition. Then, bi-temporal dense point clouds are reconstructed. Based on the area of change associated with soil collection, the constructed dense point clouds are compared according to the topographic characteristics of the area to estimate the volume of soil collected. The proposed method is stable and reliable and can meet the requirements of actual measurement tasks.

In a research for measuring package volume with multi source vision Kinect sensor.[17] The foreground segmentation and the algorithm of minimum area bounding rectangle are used to compute the package dimensions. The height of package is obtained by background subtraction method. Finally, the package volume is calculated by integration method. The experimental results show that the relative error is less than 4.10%, which improves the efficiency and accuracy of package volume measurement. The method has the features of noncontact and non-stop run. It can be adapted to different applications. The method requires only one Kinect sensor the camera parameters are calculated by using MATLAB Camera Calibration Toolbox. Calibrating process was implemented to achieve the intrinsic matrix and extrinsic parameters of the color camera of Kinect. The image processing part includes image registration and target region segmentation. The registration of image pairs was pre-processed firstly by using relative functions provided in Kinect for Windows Software Development Kit (SDK). The color images are transformed to grey images, and then the grey images are turned into binary images by finding the optimum global threshold which is obtained by using Otsu algorithm. The morphology operations used to remove the noise in the image.

In a research paper, microscopic imaging is used to explore the structure of atomizing sprays.[18] Image processing technique is developed to extract volumetric information without a need for matching. The spray is visualized using high speed long-distance microscopy from two perpendicular angles. The developed internal script employs the principle of image discretization, where each image is divided into a number of slices and the individual slice from each camera is matched to compute the liquid volume fraction in each image. The volume of individual objects is calculated based on their planar area and orientation. An error analysis is performed using dozens of three-dimensional virtual models of fragment like shapes with known volume.

A research paper introduces MUSEFood [19] to calculate food volume using data collected from multiple sensors on smartphones. MUSEFood uses FCN and utilizes shape information of food containers through multi-task learning structures, resulting in more accurate and faster image segmentation. Researchers used the MLS ranging instead of using reference objects, which improves the convenience of use and achieves higher food volume estimation accuracy. The whole task of food volume estimation is divided into three steps: Sensing, Data Processing and Data Aggregation. Smartphone camera is used to take images. The speaker of the smartphone emits Maximum Length Sequence of a specific length while the user is taking photos. Then the microphone receives the echo of the signal. For data processing, the food in the image is segmented from background. The echo signal is analyzed to calculates the vertical distance from the smartphone lens to the food. In Data Aggregation, processed data from images and sound waves are combined to build food model and finally estimate food volume.

A work aimed to estimate volume of tomato varieties grown in Turkey, by image processing techniques was performed in 2018.[20] It uses five different images of a tomato captured by high resolution digital cameras. Volume of the fruit is calculated by estimating horizontal and vertical distance of captured images. The results are validated with experimental results. The main purpose of this study was to make fast and cheap determination of the fruit quality evaluation process without damaging the fruit and making it ready for packaging. The volume of each tomato was calculated by considering ellipsoid shapes. The components of the system consist of five high resolution cameras connected to a PC via USB port, a 50 Watt LED light source, and a sample holder. Each tomato was placed at the center of the field of view and five RGB color images (1280×960 resolution) were captured and saved in jpg format. The tomato was segmented from the background using simple thresholding method combined with morphological operations in each image.

Morphological operations like dilation and filling of holes were performed on images to improve the segmentation. Otsu thresholding was used to binarize image and then remaining noises was removed. Outline of tomato was determined by edge detection functions such as Sobel, Canny.

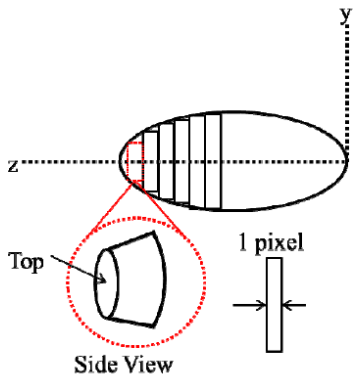


Figure 2.1 Illustration of volume calculation by considering object as an axisymmetric. The objects were modelled as a sum of conical frustums. The major dimension axis of the object and pixel height are taken as the z-axis and 1, respectively [21]

The experiment results showed that Canny edge detection methods provided better results compared with Sobel operator. Major and minor axis of each captured image were determined by built in function (region props). Euclidian distance of vertical and horizontal distances of images were calculated and compared with region props function output. Binary image was divided into pixels. The vertical and horizontal dimensions of tomato were measured using a caliper and the calibration factor was calculated as the ratio of the dimension in centimeters and the number of pixels. Calibration factor of vertical and horizontal axis was used for volume calculation.

2.10. Summary of literature review

Estimation of bulk volumes have been a topic of interest of researchers due to its applicability in engineering and businesses. With the growth of technology, the methods and approaches of bulk volume estimations are also growing. The target of new techniques always remains to achieve maximum ease of use and minimum cost with accuracy.

Numerous methods were reviewed for this thesis. Some of the limitations of the current techniques are listed below in order to get a better direction for current thesis work.

- All bulk volume measurement solutions require high computaional need in order to achieve better accuracy.

- Volume calculation solution through mobile devices involve substantial user intervention.
- As far as accuracy is concerned, only photogrammetry does not serve the purpose, additional technologies are also incorporated to facilitate the measurement for example, laser, Lidar, structured lighting, acoustic echo, etc.
- Camera calibration is always a challenge to achieve volumetric measurement through imaging.
- Extraction of ROI from an image typically depends on lighting conditions
- Accurate Volumetric Calculation through imaging depends on camera position and orientation information at the time of image capture which is difficult to achieve in mobile portable devices.

All problems stated above need further exploration. In this thesis the study is conducted to develop a solution to compute volume of bulk materials on Android devices with better accuracy and minimum user intervention.

3. METHODOLOGY

The chapter will cover the explanation of methods and approaches which have been implemented to design the solution for Bulk Volume Calculation.

3.1. Development Environments and Platforms

Development Environments and platforms used in this work are described below.

3.1.1. Android Studio

Android Studio is the official IDE for Android operating system [22] built on JetBrains' IntelliJ IDEA software and designed specifically for Android App development. It is available for download on Windows, macOS and Linux based operating. It has replaced formerly used Eclipse Android Development Tools (E-ADT) as the primary IDE for native Android application development.

Android Studio supports the programming languages Java, C++ and Android Studio 3.0 or later supports Kotlin. Android Studio features Android Emulator tool which can be installed by selecting the Emulator component from SDK tools. The Emulator simulates Android devices with almost all the functionalities of physical mobile device on your computer. This makes it very convenient to check the application with all device models and API levels without needing to have these devices physically available.

Android version and hardware characteristics of the android device used as emulator is specified through AVD Manager in Android Studio. Several devices and models can be added in the AVD manager and user can select from these models while testing the App. In the same way the App can be run and tested on a physical device by connecting it with computer system running the Android Studio. The connection can be made through USB Cable. There are several tools also available to connect the Physical Android Device with Android Studio with an over the air connection.

3.1.2. OpenCV

OpenCV stands for Open-Source Computer Vision is the most widely used Library for image processing applications. It is a computer vision and machine learning software.

OpenCV has more than 2500 optimized algorithms, including conventional image processing computer vision and state-of-the-art machine learning algorithms. These algorithms are widely used in image classification, objects detection, face detection, motion and action detection in videos, camera and moving object tracking, 3D model extraction, image stitching, 3D point cloud production, image correlation and preprocessing for augmentation reality applications. OpenCV has more than 47 thousand users and more than 18 million downloads [23].

3.2. Android Camera API

The process of Bulk Volume Calculation begins with acquisition of image of object for further processing to extract the parameters. The Android framework support three main APIs to enable various functionalities of on device cameras. The three APIs name Camera, Camera2 and CameraX. We have used CameraX API which is a Jetpack support library. It provides a consistent and easy-to-use API surface that works across most Android devices, with backward-compatibility to Android 5.0 (API level 21).

CameraX is most recent and includes all features of predecessor APIs along with resolving device compatibility issues reported in previous APIs. CameraX API reduce the amount of code needed for adding camera capabilities to the application. CameraX API can provide all capabilities of Android preinstalled camera App. Optional add-ons can enable the app with additional effects including HDR.

3.3. Segmentation

After capturing the image, we need to extract the Region of Interest from the image. The process of extracting the object of interest is termed as segmentation in image processing. There are several segmentation techniques used for wide range of machine learning and AI applications.

HOG Descriptors
Convolutional Neural Networks
Color Filtering

3.3.1. HOG Descriptors

Histograms of oriented gradients is one of the methods used for image segmentations. It works on an indicative description characterizing the shape of the object. At start, this method was used for the detection of people in images, later studies demonstrated its effectiveness for a range of classification and segmentation problems. HOG decomposes an image into small regions called cells, i.e 8x8 pixels. Then for each cell it computes a histogram of oriented gradients, the result is normalized with a block-wise pattern and a descriptor for each cell is created.

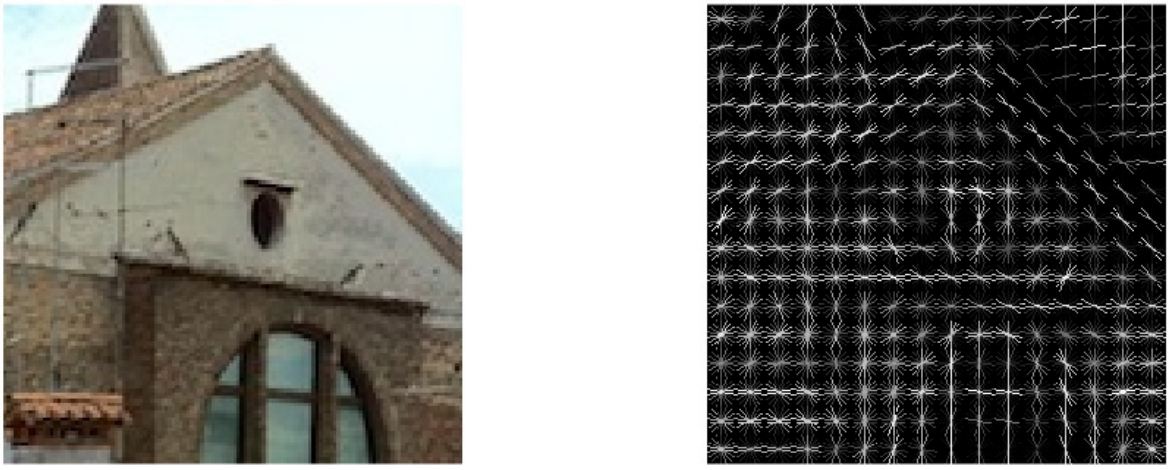


Figure 3.1 Input Image with Standard HOG Features with a cell size of eight pixels

The feature representation method Histogram of Oriented Gradients (HOG) is used with Support Vector Machines (SVM). Feature extraction from the object image is conducted by using HOG. Identity of the object is sorted through the learning process of SVM classifier. There are some limitations of the approach of segmentation through HOG Features enlisted bellow.[24]

- Hough transformation leads to the huge computational cost
- Algorithms have a high sensitivity to the noise and distortion of the target objects

Due to the reported weaknesses the author decided not to proceed with segmentation through HOG features.

3.3.2. Convolutional Neural Networks

Convolutional Neural Network also abbreviated as ConvNet is a class of deep neural network which is most commonly used for image processing. It extracts features directly from pixel images. CNN is also able to recognize patterns or features which are new to the system in case if it resembles to one of the patterns included in the training of the model.

CNN requires small amount of pre-processing as compared to other image classification and segmentation algorithms. The network is able to optimize the filters through automated learning. CNN needs less human intervention in feature extraction which is a prominent advantage.

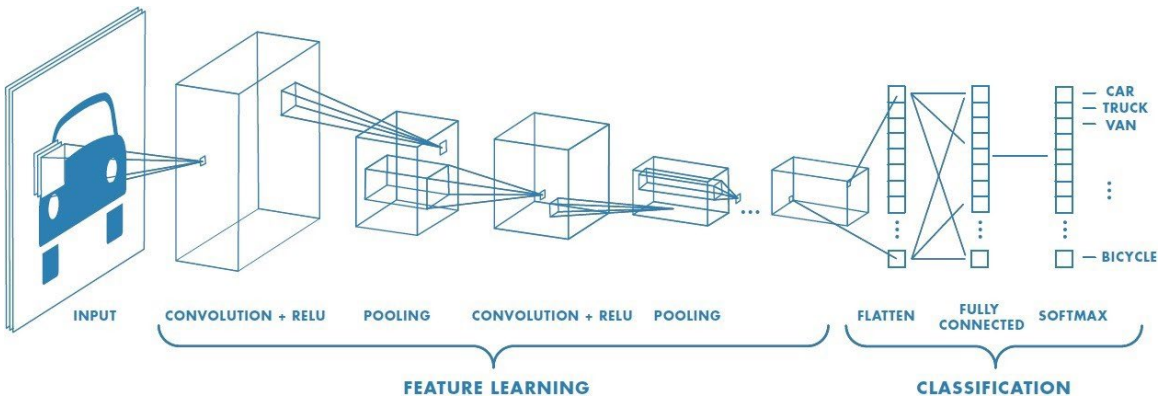


Figure 3.2 Basic Structure of Convolutional Neural Network [25]

Deep learning algorithm simplifies the process of feature extraction through a multi-layer convolutional neural network (CNN). CNN aims to transform the high dimension input image into low dimension but higher in accuracy semantic output.

Numbers of deeper and more complicated networks are developed to provide higher accuracy in computer vision applications, such as classification, detection and segmentation. As the accuracy increases the computational cost of implementation also increases. The proposed solution is aimed to be implemented on mobile devices the computational budget for the application was taken into consideration.

3.3.3. U - Net

After careful review of related work on image segmentation the author selected U-Net for creating a CNN Model for Image Segmentation. U-Net is a Convolutional Neural Network which has the advantage of low computational cost and can be trained with small dataset and offers a better accuracy. It was initially developed for biomedical image segmentation at the Computer Science Department of the University of Freiburg.[26]

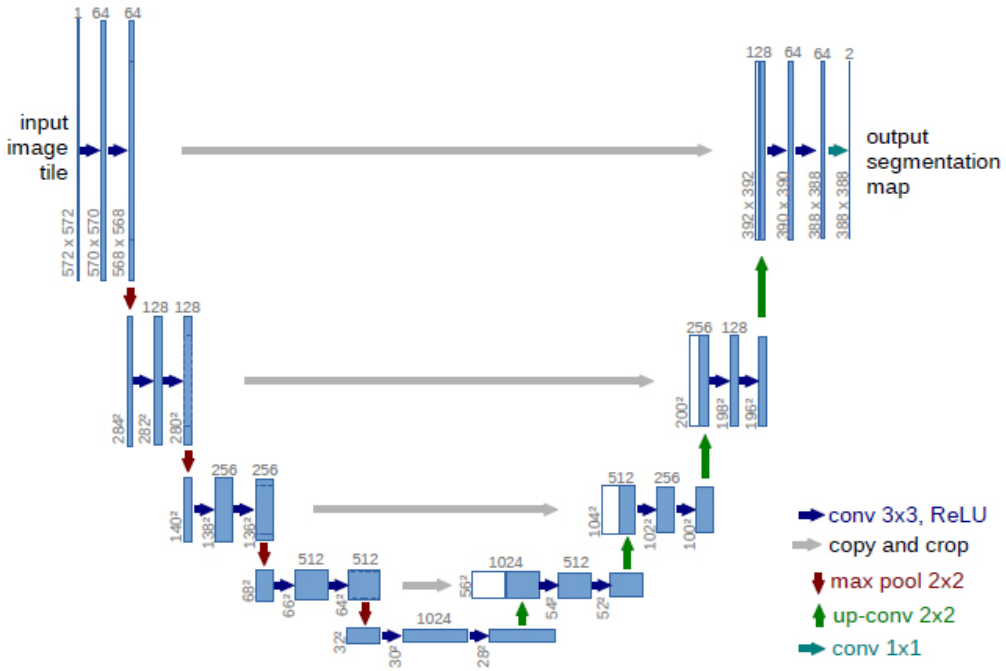


Figure 3.3 U-net architecture (example for 32x32 pixels in the lowest resolution).[26]

Figure 3.3 explains the basic structure of U-Net architecture. each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The width and height of image is provided at the lower left edge of the box. Copied feature maps are represented by the white boxes. The colored arrows in the figure denote operations described in lower right corner of the figure.

3.3.4. Transfer Learning

Transfer Learning is applied when there for training a model with relatively smaller data set. A pretrained model trained with larger dataset is used to train the new model. Transfer Learning is a Machine Learning technique in which a model developed for one task is re-used on a second related task.

In this thesis work a pre-trained MobileNetV2 is used as the encoder for the UNet architecture. MobileNets is considered one of the most efficient networks for mobile and embedded vision applications. MobileNet architecture uses depth wise separable convolutions to build light weight deep neural networks.[27]

The MobileNetV2 used in this work is trained on the ImageNet dataset which is a large visual database designed for visual object recognition software research purpose and comprise over more than 14 million hand-annotated images. We have integrated the pre-trained MobileNetV2 with the U-Net to work with smaller dataset to have an efficient network architecture.

3.3.5. Dataset for image segmentation

Segmentation accuracy altogether depends on segmentation algerithm and size of the training data set. Nevertheless, the collection of a large training data set with ground truths is a challanging task. In the proposed work author has adressed this challange with selection of U Net algorithym which has its popolarity for better accuracy with smaller datasets and Data set enhancement techniques such as Data Augmentation.

3.3.6. Data Augmentation

Arrangement of large number of images for our dataset is a time-consuming task. To overcome the limitations of small datasets we have used Data augmentation technique to increase the number of images and their perspective ground truths. Data Augmentation improves the performance of segmentation model in which set of transformations are applied in data space and/or feature space on both the image and the segmentation map. Typical transformations include flipping, rotation, scaling, cropping and projections. Data augmentation enhances the generalization and

decreases the chances of overfitting. Data augmentation has shown up to 20% improvement in model performance in some small datasets.[28]

3.3.7. Color Based Image Segmentation

In digital Image processing the image is processed using pixel values. Each pixel of the image can be classified through some basic characteristics. Color space is one the prominent attribute of pixel value in mid level image processing techniques such as segmentation. Color space can also describe the ways in which human color vision can be modeled. The basic form of image representation in computer vision is RGB Color space. There are many color spaces like BGR, NTSC, YCbCr, HSV, CMY, CMYK and HSI. An RGB image can be converting it into any other color space by means of any transformative functions.

RGB (Red, Green, and Blue) is I x J x 3 array of color pixels. These three color components can be understood as a stack of three individual layers. Every pixel of an RGB image will have a red layer, blue layer and green layer that will result in a RGB image. Value of each color in RGB can be range from 0 to 255. where (0,0,0) is representation of black and (255,255,255) represents white.

HSV stands for Hue, Saturation and Value. This color space is considered as closest to RGB color space and humans color sensations and perception. Hue is the dominant observed color. Saturation is the amount of white light varying with hue. Value is the intensity or brightness. HSV values are in range of Hue (0..359°), Saturation (0..100%) and Value(0..100%).

The HSV color space is different from RGB color space because it separates out the Intensity (luminance) from the color information (chromaticity). Difference in Hue of a pixel is visually more prominent as compared to that of the Saturation. For each pixel Hue or Intensity is chosen as dominant feature based on its Saturation value. RGB features blurs the distinction between two visually separable colors by changing the brightness. While, the HSV based segmentation is more efficient to determine the intensity and shades near the edges of an object which sharpens the boundaries retaining the color information of each pixel. [29]

4. DEVELOPMENT OF THE SOLUTION

This Chapter includes the details about experiments and procedures involved in the development of proposed solution. All major steps are described in this chapter along with stoppers and bottlenecks.

4.1. List of Equipment

The aim of the study is to provide a solution that will work on commonly available mobile devices without need of specialized hardware. The work was conducted with following physical devices along with Virtual devices available in AndoidStudio.

- For Testing

	Specification
Make and Model	SAMSUNG GALAXY A10
RAM	2GB
CPU	2x1.6 GHz Samsung Exynos 7884 Octa-core Cortex-A73
GPU	Mali-G71 MP2
CAMERA	13 MP, f/1.9, 28mm (wide), AF LED flash, panorama, HDR
DISPLAY	6.2 inches IPS LCD 16M colours 720 x 1520 pixels
OS	Android 9.0 (Pie)

- For Testing

	Specification
Make and Model	SAMSUNG GALAXY A21S
RAM	3GB
CPU	2 GHz Octa-core MediaTek Helio P35 (MT6765)
GPU	Mali-G52
CAMERA	48MP Samsung ISOCELL GM2 (S5KGM2) 1/2.0" sensor with Quad Bayer color filter.
DISPLAY	6.5 inches PLS TFT LCD 16M colors 720 x 1600 pixels
OS	Android 10

- For Development

	Specification
Make and Model	HP Probook
RAM	8GB DDR4
CPU	Intel Core i5
Diskspace	320 GB
OS	Windows 10

4.2. List of Software tools and libraries

Android Studio 4.2
Anaconda
Adobe Photoshop
OpenCV
TensorFlow
Keras
Albumentations

4.3. Programming Languages

Python
Java

4.4. Setting up Development Environment.

After finalizing the approach next step comes to setup an environment for android App development. Following are the major steps in detail. Android Studio is downloaded and installed. Android Studio Development Environment User Inface can be seen in figure 4.1.

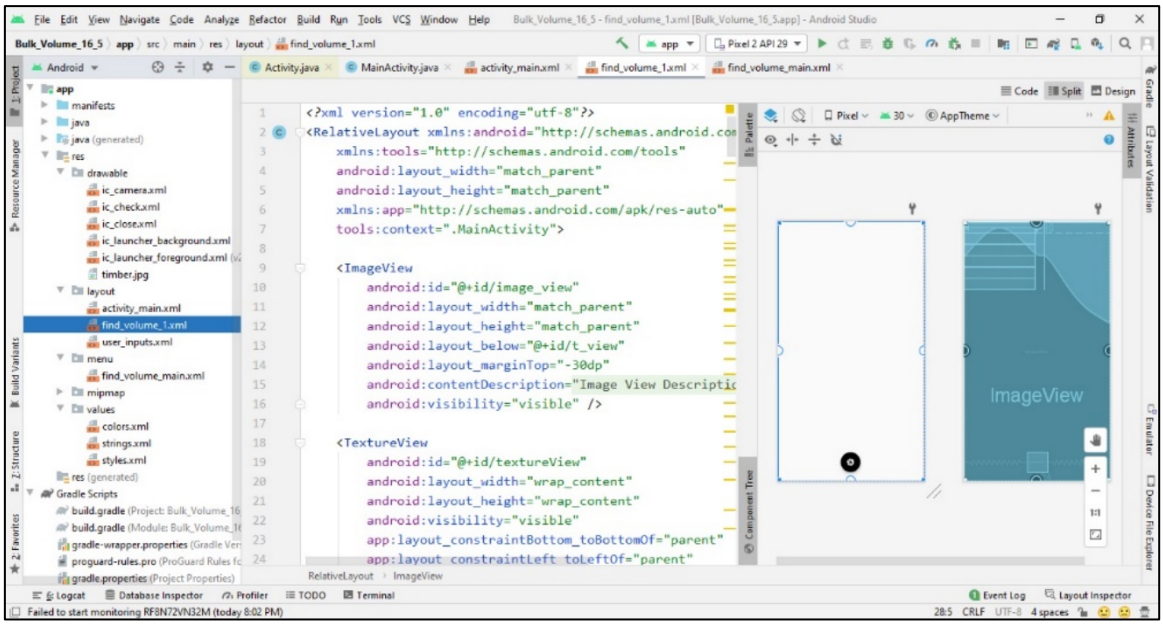


Figure 4.1 Android Studio IDE Interface.

During installation process the user is asked to include Virtual Android devices, virtual devices were used as emulators to test some of the functionalities in this work. Almost all available Android device emulators are available in Android Studio which can be configured through AVD Manager.

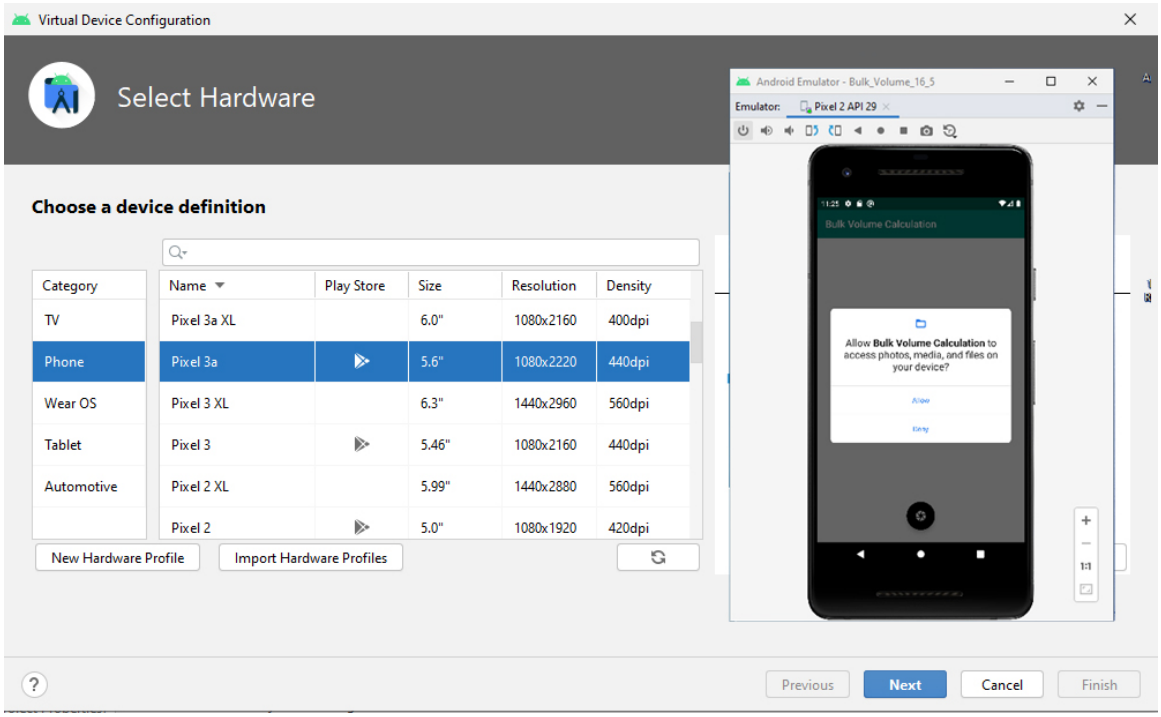


Figure 4.2 AVD Manager and Android Virtual Device Emulator

Virtual device setup also required VTx enabled in BIOS setup. Figur 4.2 shows AVD Manager and Bulk Volume Calculation App running in Emulator on Pixel 2 API 29 Android Phone. After setting up the required environment in the system the App development was started in Java.

4.5. Setting up Image Processing in Android Studio.

The very first thing for the App was to install the library for image processing. As described earlier I used OpenCV. The OpenCV library for android needs to be downloaded and imported as module in Android studio after which dependency is added in project structure. For calling the OpenCV library within the App, there are two available options.

- BaseLoaderCallBack
- Static Initialization

BaseLoaderCallBack required OpenCV Manager download when running the App. On running the App the user is asked to install the OpenCV manager application to run the App, problem here is that the OpenCV manager for android is no more available on Google Play Store, the only option is to download the APK from https://sourceforge.net/. This causes inconvenience for the user or worse can be the case if the user is with limited knowledge of mobile device operations. After reviewing several examples I used the static initialization of OpenCV. This method does not require any additional installation after installing the main Application.

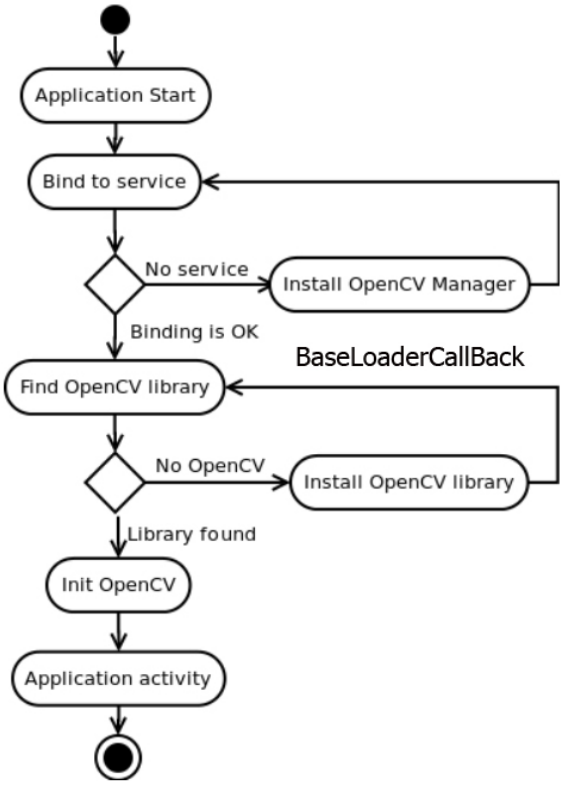


Figure 4.3 OpenCV BaseLoaderCallBack Initialization

```
static {
    if (!OpenCVLoader.initDebug())
        Log.d("ERROR", "Unable to load OpenCV");
    else
        Log.d("SUCCESS", "OpenCV loaded");
}
```

Figure 4.4 OpenCV Static Initialization log messages

4.6. Image Acquisition

Very first stage of Application Activity is to capture an Image. To capture an image in any device involves use of built-in Camera. The permissions are needed for any application to use such resources in Android platform. We specified the permissions in Manifest xml file.

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
/>
```

Figure 4.5 Camera and Storage Permissions specified in Manifest.xml

When the App checks the required permissions on start if the permissions are not already granted "`ActivityCompat.`*`requestPermissions`*" method is called and the user sees a dialog to allow the permissions shown in emulator in figure 4.2. After having all permissions, the app loads its first screen layout.

Figure 4.6 Application User Interface Image Capture Screen.

As can be seen in the Figure 4.6 the first layout/screen of the App comprises over a camera view and a capture button. The user is required to take the picture of object (log pile). After capture button is clicked an `onClickListener` captures the camera view and the user is provided with the second screen. This screen gives user two options Accept or Reject the Captured image. On clicking the Accept the preview image is send for further processing. While on reject option clicked the current camera view is resumed back and the user can take another image.

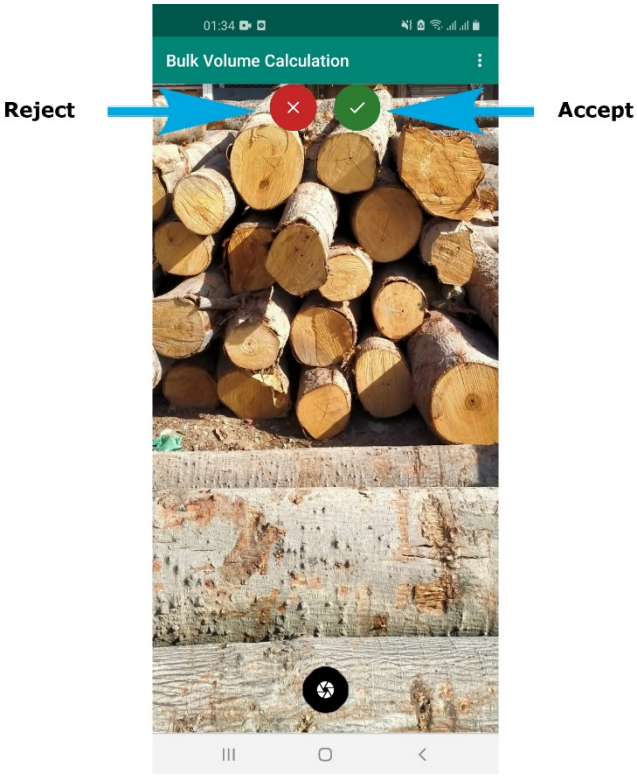


Figure 4.7 Image Capture Accept / Reject Option

After image acquisition next step is to segment and extract the onscreen area of desired object. For segmentation with CNN based approach we needed a dataset of images to train our model. Next section will describe the process of dataset creation.

4.7. Creation of Dataset

Once the image of the scene is acquired through Camera API we need to extract the localization of object of interest within our image through segmentation. A small date set of 100 images (256 x 256) is developed for this thesis which was later increased to several folds through data augmentation. The images are arranged through wood photography and online resources. The masked label for each image was created with Adobe Photoshop.

Figure 4.8 Images from dataset created for this thesis work (top row original images , bottom row perspective label masks)

4.8. Augmented Dataset

From the dataset of 100 images developed for this thesis, 600 images were created through data augmentation. The Augmentation is performed with Albumentations which is a Python library for image augmentation. The transformations used for augmentation were CenterCrop, Rotation, Grid Distortion, Vertical and horizontal Flip

```
aug = CenterCrop(128, 128, p=1.0)
aug = RandomRotate90(p=1.0)
aug = GridDistortion(p=1.0)
aug = HorizontalFlip(p=1.0)
aug = VerticalFlip(p=1.0)
```

Figure 4.9 Transformation performed during Augmentation with Albumentations.

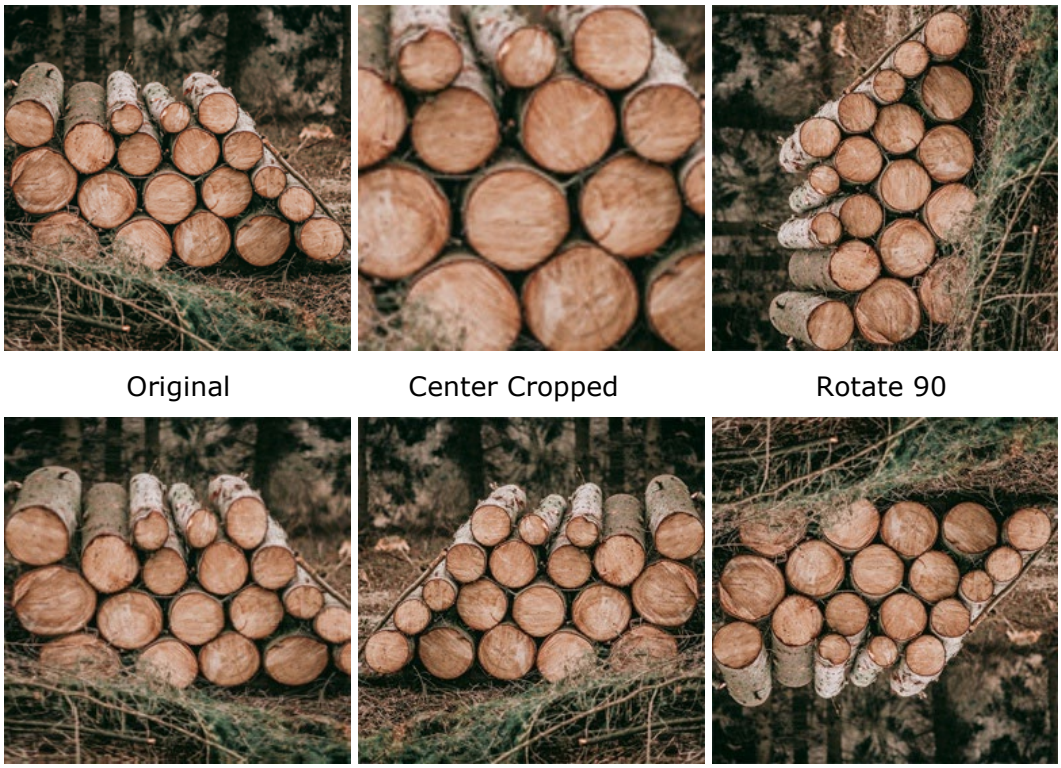


Figure 4.10 Augmentation result of the original Image.

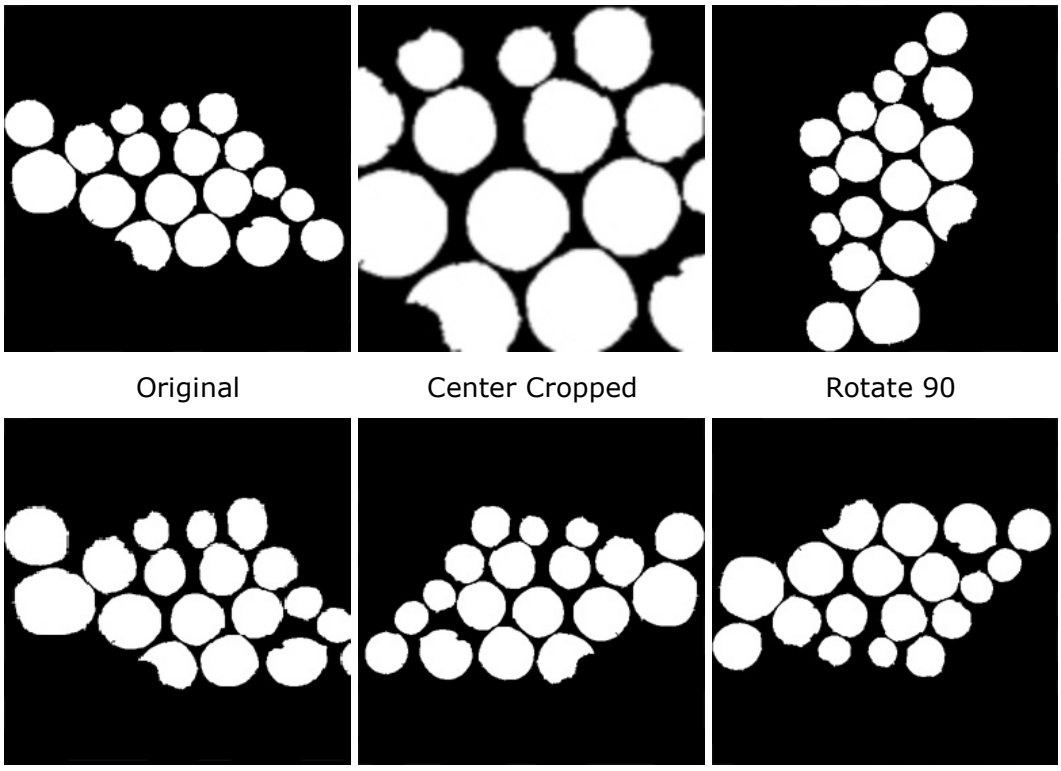


Figure 4.11 Augmentation result of the Masked label.

4.9. Segmentation Algorithm

As described in chapter no 3 we have used U-Net architecture with pre-trained MobileNetV2 as the encoder to achieve the segmentation. MobileNets is computer vision model for TensorFlow which is used for acheiving high accuracy on mobile device or embedded application. MobileNetV2 is a very effective feature extractor for segmentation and object detection. MobileNetV2 provides a very efficient mobile-oriented model that can be used as a base for many visual recognition tasks.

The segmentation model is developed in python using TensorFlow. TensorFlow is an open-source library for development and training of ML models. TensorFlow Lite is a product in the TensorFlow ecosystem which is used to run TensorFlow models on mobile devices. It provides on device machine learning inference with low latency and a small binary size at low computational cost. A Machine Learning model developed in tensor flow can be saved as tflite model which can be imported and integrated in mobile applications.

For Python development in this work Anaconda platform is used. Anaconda simplifies package management and deployment. Several environments can be configured on the same machine each comprising different packages according to the projects need. In Anaconda Jupyter Notebook After importing required libraries and functions we set some hyper parameters for training the model.

```
IMAGE_SIZE = 256
EPOCHS = 10
BATCH = 8
LR = 1e-4
```

The dataset i.e images and masks are loaded with Load data function and are split into three.

- Training dataset
- Validation dataset
- Testing dataset

In this work I have used 80, 10, 10 ratio. The images and maskes are loaded and normalize by dividing with 255.

```
encoder = MobileNetV2(input_tensor=inputs, weights="imagenet",
include_top=False, alpha=0.35)
```

MobileNetV2 is used as encoder pretrained on ImageNet dataset. Output of the encoder is fed to U-Net architecture and following Keras functions are used in the decoder.

UpSampling2D
Conv2D
Batch Normalization
ReLu Activation,
Sigmoid Activation

4.9.1. Training the Model with small dataset

First we trained the model with small dataset of 100 images and their perspectives masks.

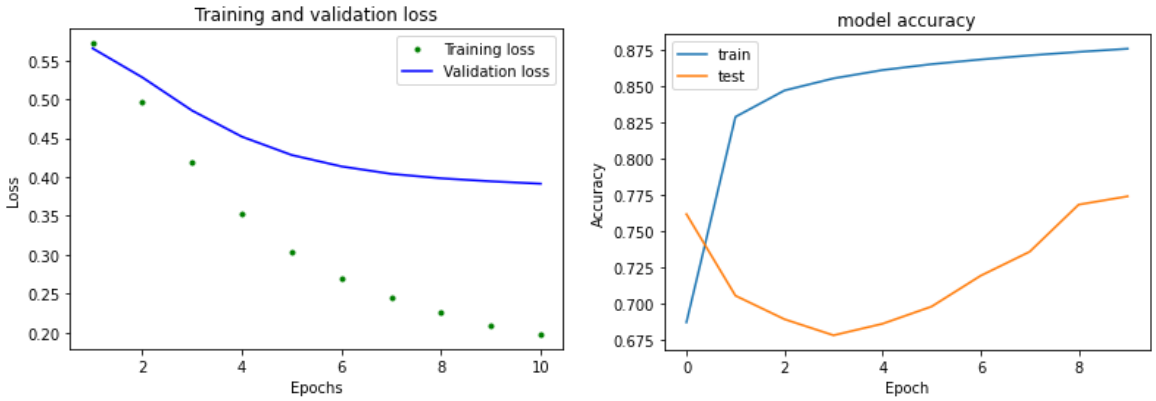


Figure 4.12 Training and validation, Loss and Accuracy plot for small dataset

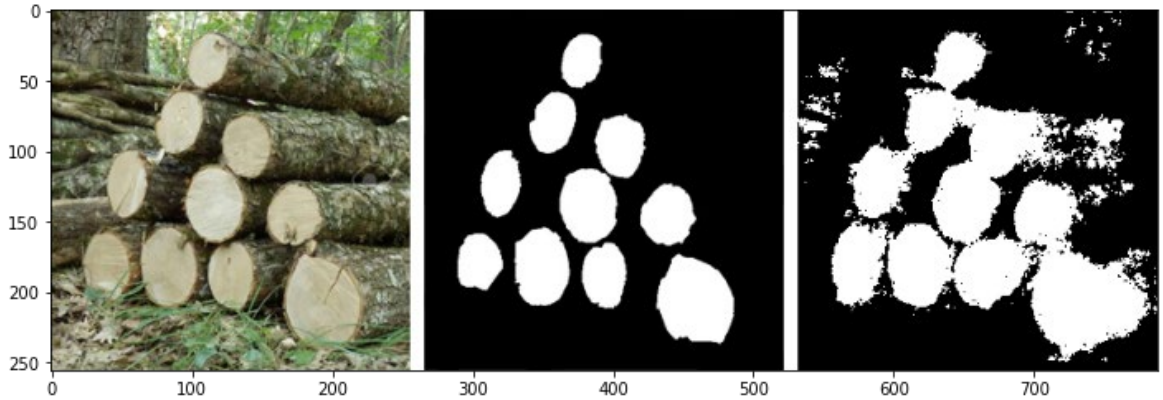


Figure 4.13 Model performance on small dataset from left to right Original Image, Perspective mask and Model output.

The model performance graph shows that with small dataset of 100 images the U-Net model after 10 epochs was able to achieve the training accuracy 0.8757 and loss was reduced to 0.1977. The data set was split with under mentioned ratio.

```
Training data:  80
Validation data:  10
Testing data:  10
```

4.9.2. Training the Model with large dataset

Data augmentation technique was applied to small dataset and 600 images were produced out of 100 images by applying image transformations. The augmentation of the data has improved the result which can be seen in the Figure 4.14 and 4.15.

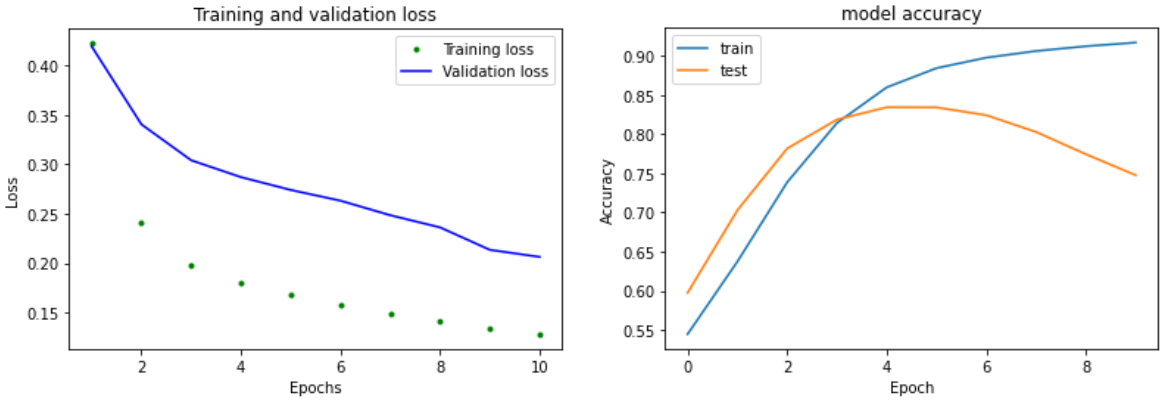


Figure 4.14 Training and validation, Loss and Accuracy plot for large dataset

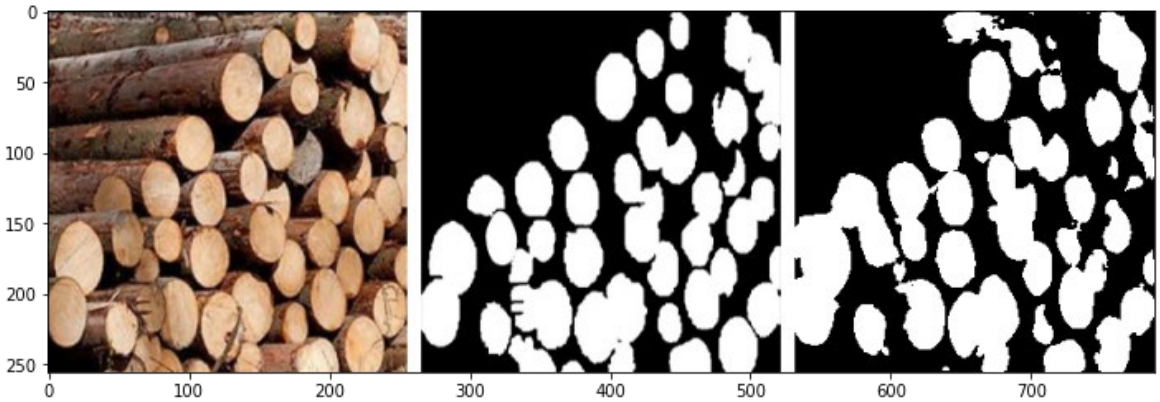


Figure 4.15 Model performance on large dataset from left to right Original Image, Perspective mask and Model output.

The training accuracy 0.9170 and loss was 0.1281. The data set was split with under mentioned ratio.

```
Training data:  480
Validation data:  60
Testing data:  60
```

After completion of Training, the model was saved as Tensoflow file and then converted to tflite model for using it in the android application.

4.10. HSV Range Selection

For segmenting the desired object from an image, we developed a tool for extracting and masking the pixels which correspond to specific object. The HSV features retain the identity of the colors even at different light intensity levels which makes the HSV-based features very useful in running segmentation algorithms. Due to the reason, In this work HSV value are used instead of RGB values for Range selection tool.

RGB to HSV conversion needs special care when working with OpenCV library. OpenCV value of Hue in HSV color space is different than other photo conversion programs. For hue value in OpenCV library we need to define it in a range from 0 to 179 while generally it is taken on a range from 0 to 359 in degrees. HSV color range tool used in this work comprises over 6 Trackbars denoting the Upper and lower range for H, S and V respectively.

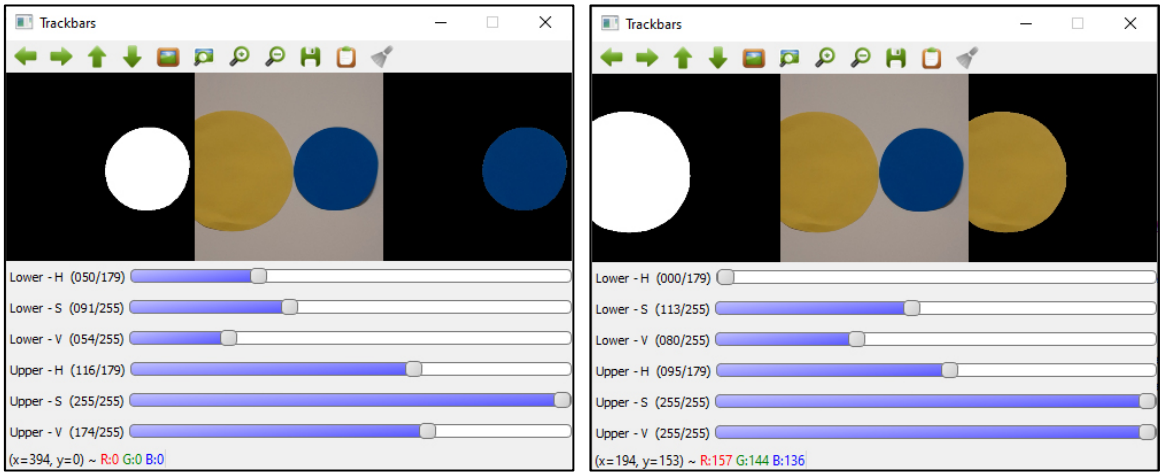


Figure 4.16 Segmentation tool range boundaries for blue and yellow color used for experiments.

This tool is developed with OpenCV 4.5.2. First, I created the trackbars and set their initial and maximum values. For Hue the range is set from 0 to 179 and the starting value for Lower H is 0 and Upper H is 179. For Saturation and Value, the Lower value is 0 and the Upper value is 255. The initial value for Lower of both is 0 and for Upper of both is 255. Then I set "cv2.getTrackbarPos" for each trackbar to take input defined by user in the trackbar window. All 6 values are used to set NumPy array to set the boundaries of the mask output. Original Image, Masked Image and Segmentation result are displayed in the staked window for comparison.

```
cv2.createTrackbar("Lower - H", "Trackbars", 0, 179, trackChanged)
cv2.createTrackbar("Lower - S", "Trackbars", 0, 255, trackChanged)
cv2.createTrackbar("Lower - V", "Trackbars", 0, 255, trackChanged)
cv2.createTrackbar("Upper - H", "Trackbars", 179, 179, trackChanged)
cv2.createTrackbar("Upper - S", "Trackbars", 255, 255, trackChanged)
cv2.createTrackbar("Upper - V", "Trackbars", 255, 255, trackChanged)
```

Figure 4.17 Defining trackbars in color segmentation tool.

In the following experiment, the tool shows the view of hsv transformed image, masked area corresponding the hsv range between lower and upper limits set by trackbars. User can adjust the upper and lower limits of HSV to get the only required object in image completely masked. In the result window the segmented object can be seen. Once the "in range" values for a particular object are achieved with help of the tool, we can test these values with similar images having possible objects in the image along with desired object and the limits can be further adjusted for fine tuning.

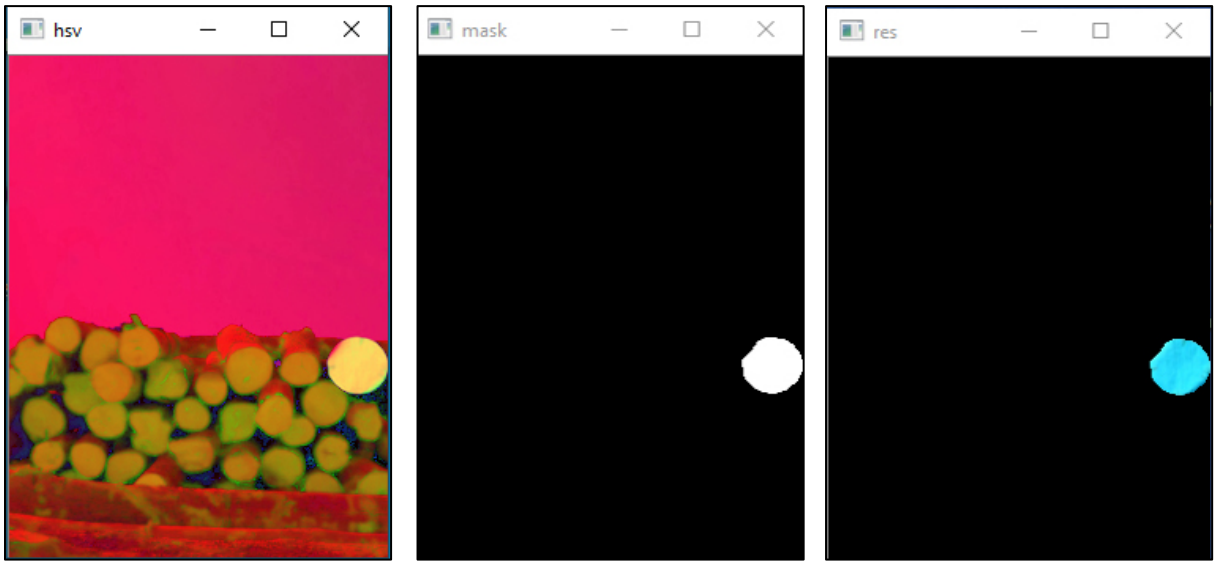


Figure 4.18 Reference Object HSV image, Mask and Segmentation result

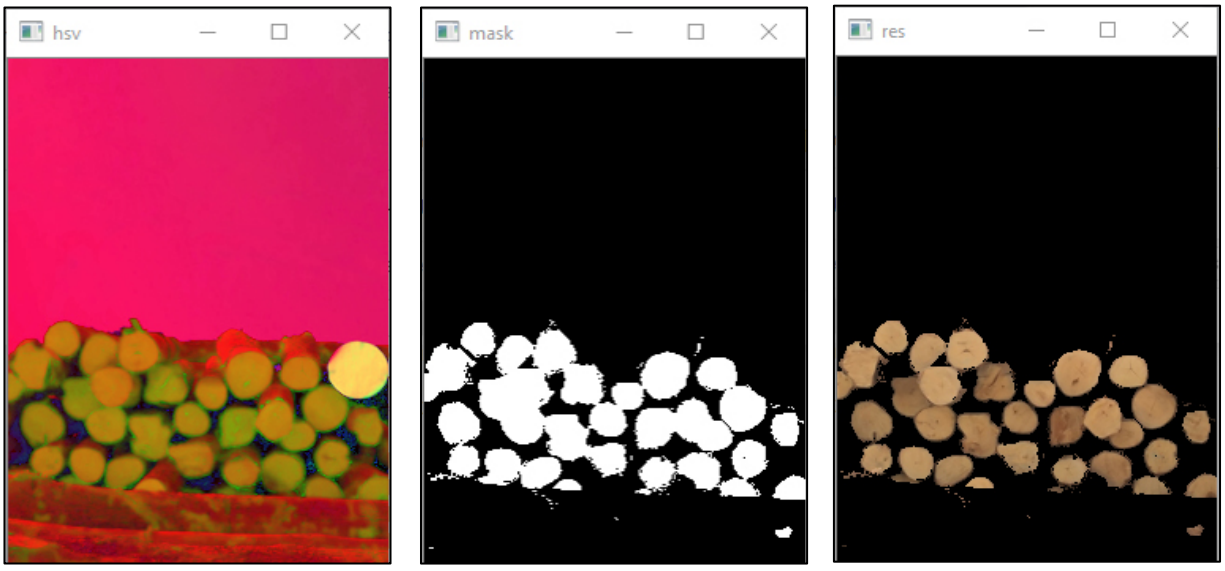


Figure 4.19 Wood logs HSV image, Mask and Segmentation result

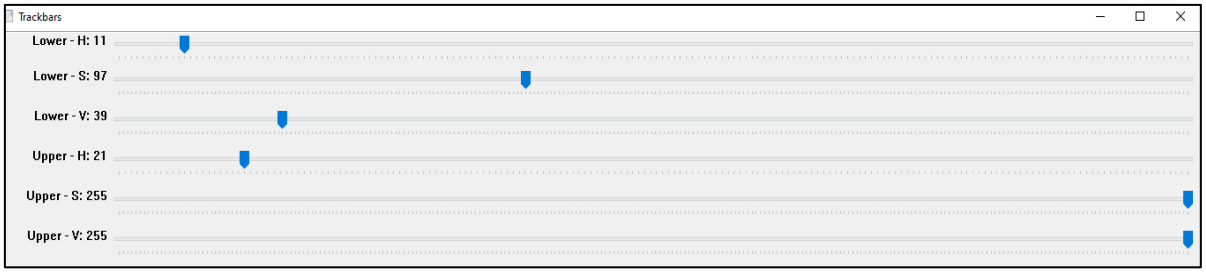


Figure 4.20 Trackbars set for Wood logs segmentation

After processing an image with wood logs and a reference object we have found following upper and lower boundaries respectively

```
Lower limit  (90, 146, 162)
Upper limit  (114, 255, 255)
Lower limit  (11, 97, 39)
Upper limit  (21, 255, 255)
```

These values will be used in the next section for color base segmentation in Bulk Volume Calculation App.

4.11. Android App Development

In this work Android Studio is used to create the App. A new project was created with following configuration

```
compileSdkVersion 28
buildToolsVersion "29.0.0"
minSdkVersion 21
targetSdkVersion 28
```

Language is set to java. All required libraries were added to the application. As mentioned earlier we have used CameraX API for capturing the image. OpenCV was added to dependencies and initialized in the App.

App layout is created which contains a Texture View to show the preview of the camera, and capture button which allows to capture the image on click. The captured image is set to be displayed in an Image view.

On receiving a click on the accept button the application segments the image on provided HSV Values and calculates the pixel values for objects. The user is prompted to insert the known parameters of the object through a popup window. The captured image is in the bitmap format, It is converted to Mat for further processing. Following methods are used for bitmap to mat conversion and to HSV conversion.

```
Utils.bitmapToMat(capturedBitmap, originalMat);
Imgproc.cvtColor(originalMat, hsvMat, Imgproc.COLOR_RGB2HSV);
```

After converting to HSV the image is segmented with "`Core.inRange`" function which is simple thresholding function of OpenCV. In this function the ranges found from Segmentation tool are fed in parameters and a Mask in created.

```
Core.inRange(hsvMat, timberlowerRange, timberupperRange, timberMask);
```

After creating the mask, volume can be calculated. I experimented with two methods of finding volume in pixels, finding contour area and secondly with pixel count. Drawing contours considers adjacent edges as single contour and when dealing with object like wood logs we have to consider empty area between logs. Simple drawing contour was considering the empty area as timber and hence was giving wrong value. Although there are techniques like method of circle detection with Hough transformation but these are sensitive to noise and variation in distance of object. Drawing contours was found effective for reference object because it had a single big contour. For wood logs I used Pixel count method as shown below.

```
Core.countNonZero(timberMask);
```

With the segmentation the only area (pixels) which corresponds to timber is extracted and a gray scale mask is created on the Area. Counting non zero pixels in the Mat of this mask gives the correct area in pixels.

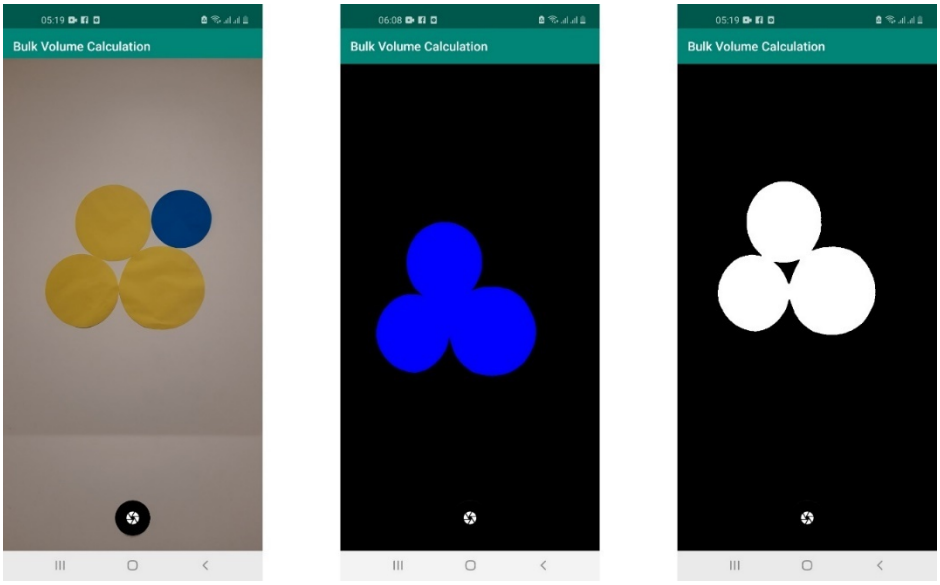


Figure 4.21 From left to right- Original image, contour area mask, segmentation mask

Find contour function also makes all the contour visible along with unwanted noise. To remove the noise contours are filtered on the basis of Contour Area. The area being measured is also made visible to the user in order to check if the segmentation is correct or there is a need for taking another measurement. When the App finishes segmentation and finding surface Area of reference object and measured object, the App requires the input from user to convert the pixel values to metric units.

Following method is used to call "User Input" fragment to display dialog on screen. Dialog layout is defined in the xml file under layout folder. The input fields are created "edit text" in the layout.

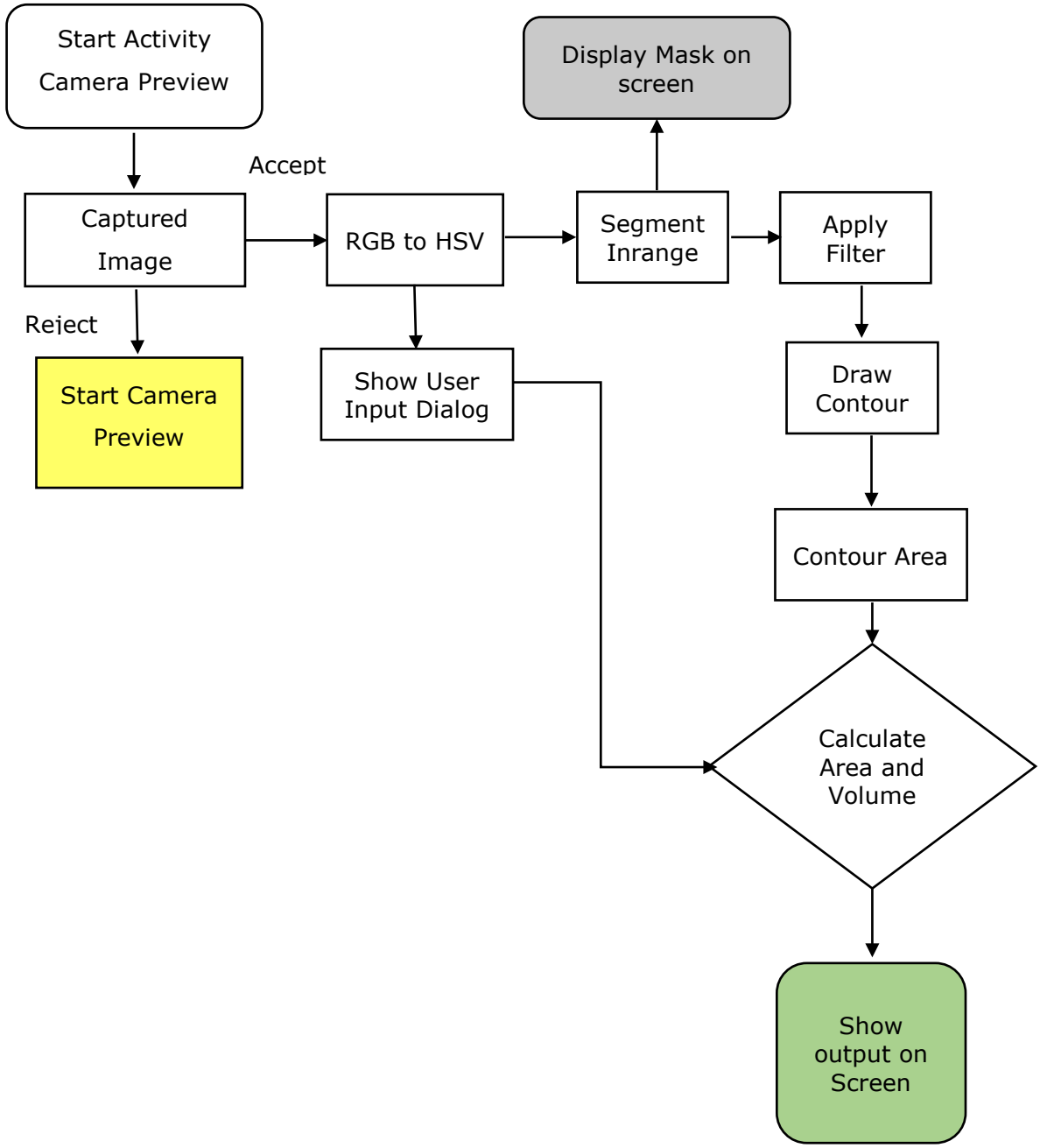
```
private void openDialog() {
    UserInputDialog userInputDialog = new UserInputDialog();
    userInputDialog.show(getSupportFragmentManager(), "User Inputs");
}
```

The user is asked to provide the surface area of the reference object and the length of the logs in pile. With the help of user provided information after making calculation the App displays the surface area of all detected logs and volume of the pile. At this stage we faced a problem the edit text gets the input as "string" but arithmetic calculations can not be performed on strings. Following conversion is used.

```
double length = Double.parseDouble(log_Length);
```

After getting the "double" values simple arithmetic calculations are performed to find the volume. Final Calculation is explained in Section 4.13

4.12. App Flow chart



4.13. Volume Calculation and Evaluation of Result

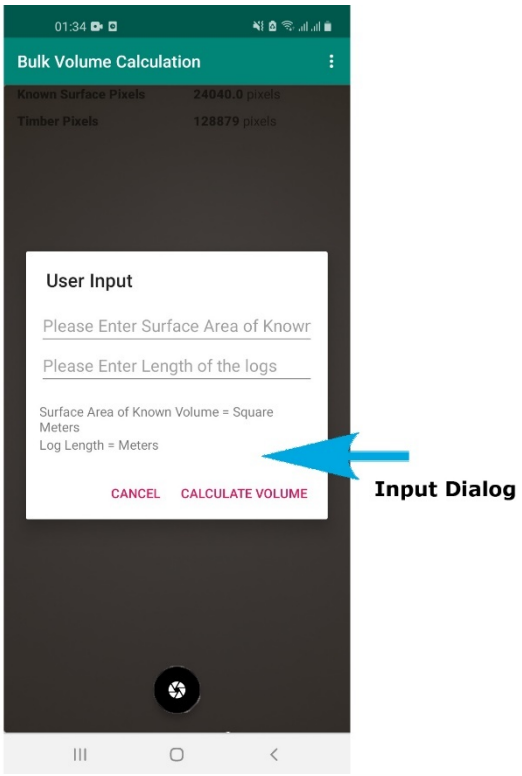


Figure 4.22 User Input Dialog for known parameters

After capturing and accepting the image. The dialog box appears on screen which needs two inputs. Surface Area of the known reference volume and the length of the logs in the pile being measured. The app was first tested on color circles pasted on a wall. Two colors were used in this experiment, Blue for reference objects and Yellow for object to be measured. First we find out the HSV range for these two colors with the help of HSV color range tool.

```
BluelowerRange =        (50, 91, 54);
BlueupperRange =        (116, 255, 174);
YellowlowerRange =      (0, 113, 80);
YellowupperRange =      (95, 255, 255);
```

These ranges are set in the App for segmentation. The surface area of the reference objects was measured with manual method.

$$V = \sum_{i=1}^{N} \left(\frac{d_i}{2}\right)^2 l \tag{4.1}$$

Equation 4.1 shows the volume calculation for log pile with conventional method, where V is total volume, i is number of log, d is diameter or average diameter, l is the length of the log.

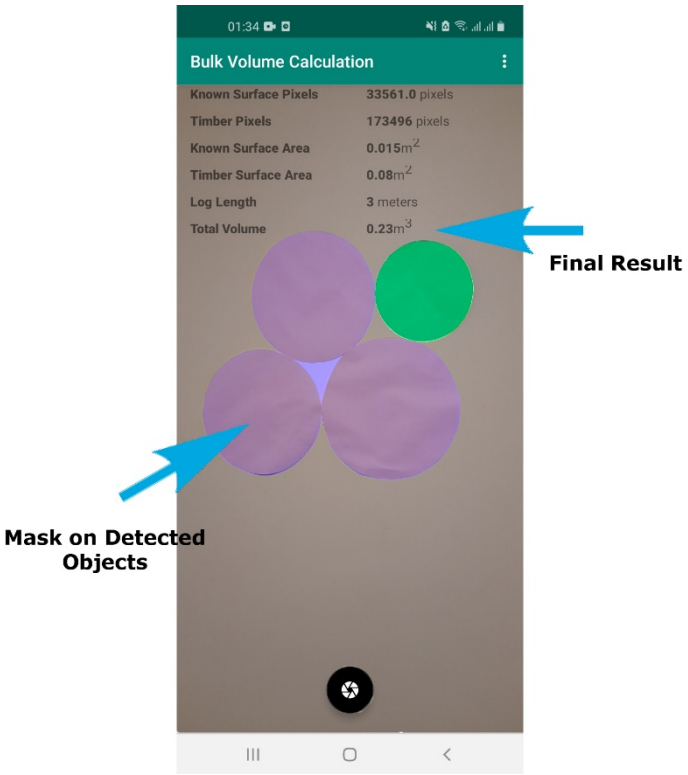


Figure 4.23 Final Calculation Experiment Result

Through user input the App has provided the surface area of known object which is blue circle in this experiment.

The diameter of the reference object was measured 14 cm which makes the 0.015 m² Surface Area. Like wise to compare the result we took diameters of Yellow circles and calculated the surface area with equation 4.1.

Diameters = 0.21, 0.185, 0.19 meters
Surface Area = 0.031, 0.025, 0.028 m²

Total Surface Area calculated with manual method = 0.084 m²

Total Surface Area calculated with Developed App = 0.08 m²

Total Volume calculated with manual method = 0.252 m³

Total Volume calculated with Developed App = 0.23 m³

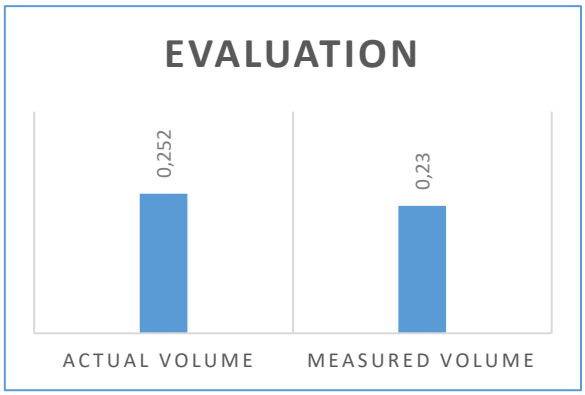


Figure 4.24 Experiment Result Comparison

For ease of understanding we have rounded the outputs in two decimal places and default measurement unit is meter. From the result it can be seen that the accuracy of the measurement is quite good.

5. DISCUSSION

In this chapter, the thesis work outcomes are summarized along with limitations of the work result and future directions on the subject matter. Achieved objectives of the work are discussed in this section and suggestions are outlined for improvement in accuracy and convenience in use.

5.1. Limitations

As the thesis topic covers a wide range of researchable and continuously improving areas the limitations are also copped at a proportional rate. Few of the prominent limitation are enlisted bellow.

- The Accuracy of the system depends on certain work conditions. The reference object should be placed in the same vertical plane of wood abuts. The images should be taken by keeping the camera in upright positions. Images taken from tilted camera causes reduced accuracy. Maximum accuracy is achieved through taking image by keeping the pile in center of the screen.

- In Android App development when working with third party applications unprecedented errors are occurred on which most of the times the official support platforms also do not have adequate guidance. Online resources on the subject are insufficient and require years of practice while on the other hand deprecated classes and unstable updates make the progress even slower. Several issues escalated on OpenCV and TensorFlow official forums are still unanswered. This makes development work difficult and one has to give a try to number of solutions provided by other developers which is time consuming.

- Tensor Flow lite models once generated on python was found to be not working with the developed Android App due to which the segmentation was performed on the HSV color-based segmentation in order to demonstrate the working of later stages of the Application. Although all instructions on the tflite support page and demo were followed.

- CNN based models for segmentation need larger datasets due to its dependence of accuracy on the size of dataset. Some of the popular databases for similar researches include thousands of images i.e The Hawkwood databse consist of 7655 images. [30] In the research work, the author of the study claims that the

data set to be publicly available but specified web links are not functional. Creating such a large dataset along with perspective labeled masks can take months or even years.

- Testing of such a solution on real scenes with variance in the surrounding was also a daunting task. It was very difficult to spot such sites near the city with number of log piles. Nearest presence of such piles was located on Lemeks Sadamad AS, 45 KM of drive from the university and access into the port area required special permissions.

- Testing of an Android Application is one of the most important phases of development. The said Application required camera functionally that's why testing on Emulator devices was not very helpful. The testing of the App is performed based on available sources during which intermittent app crash has been reported. Although number of such crashes was negligible but stability of the App requires further work on testing phase.

5.2. Justification of the work completed.

Bulk volume measurement is one of the key activities involved in number of businesses mentioned briefly in the first chapter of this document. Development of such a solution to get the bulk volume calculations performed by handheld devices was the main objective of this thesis work. The author has developed an standalone App for Android mobile devices which is able to calculate volume of bulk materials with higher accuracy provided the reference object for measurement and common dimension. The latest CameraX App is used to develop this solution instead of using "`intent`" function to use built-in Camera App within Android platform. The basic aim of using CameraX API was to make it easier to manipulate the Camera Parameters in the future developments of the work in order to get high level image processing including stereo imaging and depth imaging with mobile phone camera.

The image processing in the developed App is performed with OpenCV which is the most widely used image processing library. Using OpenCV makes it easier for developers to access many advanced computer vision algorithms used for image and video processing in 2D and 3D. One of the high-end approaches in the future versions of the App may include these features and existing Application can be expanded in functionality through basic knowledge in OpenCV.

Two possible approaches are presented for segmentation in the developed App, Segmentation through tflite U – Net model and segmentation through HSV color range selector of the object. either of these can be integrated within the App to make it a novel solution.

5.3. Future directions on the thesis work

As per the opinion of the author the main attraction of the thesis topic is endless room for improvements in the subject area. Some of the suggestions for future work are listed bellow

First of All, the segmentation tflite model can be integrated within the App to improve the accuracy of extracting region of measured objects. The dataset can be expanded by adding more images of the wood logs. Further classes can be added in the segmentation model to expand the area of coverage on more objects detected and measured.

The HSV Color range finder can be added in the App as a fragment which will give additional customization for the App according to the use case. The App HSV lower and upper color range setting can be made user selected depending on the result of integrated range selection tool.

Image stitching and panorama can be added in the functionalities in order to get calculations for larger number of logs. Gyroscope sensor of mobile phone can be integrated in the app to ensure the orientation of the mobile phone while capturing the image which will improve the accuracy.

Use of reference object with in the image for pixel to measuring unit (meters) conversion can be substituted with some other techniques i.e stereo imaging by making research on cell phone location accuracy with built-in GPS sensors and accelerometer. There are several techniques of image processing including 3D reconstruction which compute distance of the object from camera calculated through images taken from different angles. One of the proposed approaches which needs some detailed research is disparity mapping between two images taken from same angle one at a position and the other after taking a step forward towards the target. The distance between camera locations can be estimated as proportion of person step size to height of the person (As a rough estimate its about 41 to 45% of ones height). Step size displacement accuracy can be studied by making experiments with the accelerometer readings.

5.4. Summary

Calculating the volume of bulk materials using modern portable devices was the topic of this master thesis work. There are several approaches available for solving the problem. A detail review of the existing techniques used for addressing the problem was carried out in order to finalize the approach. The work was decided to be based on close range photogrammetry technique to make it applicable for all Android portable devices. The other available approaches involved use of additional features which are not available on all portable devices which included use of Augmented Reality and Lidar. Once the approach is finalized then came the selection of Android Camera API. The alternative approach was to use Android built-in camera App. In this work I have used CameraX API which will enable the developer to adjust Camera parameters in further expansion phases of the Application. Prominent camera features which can play important role for image processing include manual Focus and Shutter Speed.

The image acquired by Camera is fed to App process where pre processing is performed and the image undergoes segmentation to extract the region of interest. There are two segmentation techniques used in this work one relies on Convolutional Neural Network and the other on Color based segmentation. U-Net algorithm in CNN based method is used for image segmentation along with MobileNetV2 model trained on ImageNet dataset at encoder stage. U-Net is known for its better accuracy with small datasets and MobileNetV2 is known as one of the best approaches for embedded systems and mobile devices. A data set of 100 images was created for this project along with their perspective label masks. We first trained the model with 100 images and took the observations for loss and accuracy. Afterward data augmentation is used to increase the number of images and masks from 100 to 600. The model is trained again with 600 images and the results were compared. The model was created in python using Tensorflow library, the model was saved in tflite format for integration in developed mobile App. After facing issues with integration of the tflite model in the App the parallel approach was considered for segmentation with color values. After reviewing researches I concluded to perform the segmentation based on HSV color space considering its better performance on wide range of variance in illumination. A tool for finding the HSV range of the reference and measured object was developed. After finding the upper and lower limit values of the desired objects, the values are set in the developed App for segmentation.

The android App is developed using OpenCV image processing functionalities. The App captures the image and segments the area of interest and calculates the surface area

in pixels. The user is then prompted to give input values for surface area of the reference object in the image and length of log piles. A mask on the segmented objects is created on the output image and the values of the calculated surface area and volumes are displayed on the screen. The testing of the App is performed on the model objects and results are compared with manual measurements with conventional methods.

5.5. Kokkuvõte

Puistematerjali mahu arvutamine kaasaegsete kaasaskantavate seadmete abil oli selle magistritöö teema. Probleemi lahendamiseks on mitu lähenemisviisi. Vaatasin üksikasjalikult läbi olemasolevad võtted lõpliku lähenemisviisi leidmiseks. Töö otsustati põhineda fotogrammeetria tehnikatel, et see oleks rakendatav peaaegu kõigi kaasaegsete kaasaskantavate seadmete jaoks. Teised saadaolevad lähenemisviisid hõlmasid lisa funktsioonide kasutamist, mis pole kõigis kaasaskantavates seadmetes saadaval. Nende meetodite hulka kuulusid liitreaalsus ja Lidar. Kui lähenemine on lõpule viidud, valiti Android Camera API. Alternatiivne lähenemine oli kasutada Androidi sisseehitatud kaamera rakendust. Selles töös olen kasutanud CameraX API-d, mis võimaldab arendajal kohandada kaamera parameetreid rakenduse edasistes laiendusetappides. Kaamera fookuse ja säriaja reguleerimine võib pilditöötluses olulist rolli mängida.

Kaamera omandatud pilt suunatakse rakenduse protsessi, kus toimub eeltöötlus ja pilt segmenteeritakse huvipakkuva piirkonna eraldamiseks. Selles töös kasutatakse kahte segmenteerimistehnikat, üks tugineb konvolutsioonilisele närvivõrgule ja teine värvipõhisele segmenteerimisele. U-Net algoritmi CNN-põhises meetodis kasutatakse piltide segmenteerimiseks koos MobileNetV2 mudeliga, mis on koolitatud ImageNeti andmekogumi jaoks kodeerija etapis. U-Net on tuntud oma parema täpsuse poolest väikeste andmekogumitega ning MobileNetV2 on tuntud kui üks parimaid lähenemisviise manustatud süsteemide ja mobiilseadmete jaoks. Selle projekti jaoks loodi 100 pildiga andmekomplekt koos nende maskidega. Esmalt koolitasime mudeli 100 pildiga ja võtsime vaatlused kaotuse ja täpsuse osas. pärast seda kasutasime andmete suurendamist, et suurendada piltide ja maskide arvu 100-lt 600-le. Mudelit treenitakse uuesti 600 pildiga ja tulemusi võrreldi. Mudel loodi Pythonis Tensorflow teeki kasutades, mudel salvestati tflite-vormingus selle kasutamiseks mobiilirakenduses. Rakenduses tflite mudeli kasutamisel tekkis probleeme. Proovisin teist värvipõhise segmentimise meetodit. Pärast uuringute kaalumist jõudsin järeldusele, et teostan segmenteerimist HSV värviruumi põhjal. parema jõudluse tõttu valguse varieerumisel. Töötati välja tööriist objekti HSV vahemiku leidmiseks. Pärast värvivahemiku ülemise ja alumise piirväärtuse leidmist seadsin need väärtused rakenduses segmenteerimiseks.

Androidi rakendus on välja töötatud OpenCV pilditöötlusfunktsioonide abil. Rakendus hõivab pildi ja segmenteerib huvipakkuva ala ning arvutab pinna pikslites. Kasutajal palutakse anda sisendväärtused viiteobjekti pindala kohta pildil ja palgihunnikute pikkus. Segmenteeritud objektidele luuakse mask väljundpildile ning arvutatud pinna ja

mahtude väärtused kuvatakse ekraanil. Rakenduse testimine viiakse läbi mudeli objektidel ja tulemusi võrreldakse tavapäraste meetoditega manuaalsete mõõtmistega.

LIST OF REFERENCES

[1] Z. Chunsen and Z. Qiyuan, "Research on Volumetric Calculation of Multi-Vision Geometry UAV Image Volume," 2018, doi: 10.1109/EORSA.2018.8598608.

[2] D. Shvarts and M. Tamre, "Bulk material volume estimation method and system for logistic applications," in *Proceedings of the International Conference of DAAAM Baltic* , 2014, vol. 2014-Janua.

[3] A. M. Samad, N. A. Asri, and A. Ahmad, "The use of digital image for volume determination using digital close range photogrammetric method," in *Proceedings - 2012 IEEE 8th International Colloquium on Signal Processing and Its Applications, CSPA 2012*, 2012, pp. 321–324, doi: 10.1109/CSPA.2012.6194742.

[4] I. H. Budianto and S. K.-E. Gan, "APD volumetric app: android app for the quantification of reagents," *Sci. Phone Apps Mob. Devices*, vol. 3, no. 1, 2017, doi: 10.1186/s41070-017-0019-8.

[5] G. Spreitzer, J. Tunnicliffe, and H. Friedrich, "Using Structure from Motion photogrammetry to assess large wood (LW) accumulations in the field," *Geomorphology*, vol. 346, 2019, doi: 10.1016/j.geomorph.2019.106851.

[6] J. Chopin, H. Laga, and S. J. Miklavcic, "A new method for accurate, high-throughput volume estimation from three 2D projective images," *Int. J. Food Prop.*, vol. 20, no. 10, 2017, doi: 10.1080/10942912.2016.1236814.

[7] A. Kruglov and E. Shishko, "Log pile measurement through 3D modeling," in *2017 40th International Conference on Telecommunications and Signal Processing, TSP 2017*, 2017, vol. 2017-January, doi: 10.1109/TSP.2017.8075983.

[8] T. R. Jadhav and S. M. Kamble, "Volume measurement of object using computer vision," in *2016 IEEE International Conference on Recent Trends in Electronics, Information and Communication Technology, RTEICT 2016 - Proceedings*, 2017, pp. 1792–1795, doi: 10.1109/RTEICT.2016.7808143.

[9] Y. Sun, T. Yang, X. Cheng, and Y. Qin, "Volume Measurement of Moving Irregular Objects Using Linear Laser and Camera," in *8th Annual IEEE International Conference on Cyber Technology in Automation, Control and Intelligent Systems, CYBER 2018*, Apr. 2019, pp. 1288–1293, doi:

10.1109/CYBER.2018.8688302.

[10] T. Peng, Z. Zhang, Y. Song, F. Chen, and D. Zeng, "Portable system for box volume measurement based on line-structured light vision and deep learning," *Sensors (Switzerland)*, vol. 19, no. 18, 2019, doi: 10.3390/s19183921.

[11] J. Li, G. Liu, and Y. Liu, "A dynamic volume measurement system with structured light vision," in *Proceedings - 2016 31st Youth Academic Annual Conference of Chinese Association of Automation, YAC 2016*, Jan. 2017, pp. 251–255, doi: 10.1109/YAC.2016.7804898.

[12] Y. Long *et al.*, "Potato volume measurement based on RGB-D camera," *IFAC-PapersOnLine*, vol. 51, no. 17, 2018, doi: 10.1016/j.ifacol.2018.08.157.

[13] A. Gao, F. P. W. Lo, and B. Lo, "Food volume estimation for quantifying dietary intake with a wearable camera," in *2018 IEEE 15th International Conference on Wearable and Implantable Body Sensor Networks, BSN 2018*, 2018, vol. 2018-Janua, pp. 110–113, doi: 10.1109/BSN.2018.8329671.

[14] M. A. Tamin, N. Darwin, Z. Majid, M. F. Mohd Ariff, K. M. Idris, and A. Manan Samad, "Volume Estimation of Stockpile Using Unmanned Aerial Vehicle," in *Proceedings - 9th IEEE International Conference on Control System, Computing and Engineering, ICCSCE 2019*, Nov. 2019, pp. 49–54, doi: 10.1109/ICCSCE47578.2019.9068543.

[15] H. He, X. Xu, T. Chen, and P. Lu, "Volume measurement of sand carrier using uav-based mapping," in *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2020, vol. 5, no. 3, pp. 19–24, doi: 10.5194/isprs-Annals-V-3-2020-19-2020.

[16] S. Ding, X. Zhang, Q. Yu, L. Li, and J. Wang, "A volume measurement method for lunar soil collection based on a single monitoring camera," *Sensors (Switzerland)*, vol. 18, no. 10, 2018, doi: 10.3390/s18103394.

[17] L. Li, X. Zhuang, L. Chen, B. Liu, and T. Wang, "An adapted vision measurement method for package volume based on Kinect," in *ICNC-FSKD 2017 - 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery*, 2018, pp. 918–922, doi: 10.1109/FSKD.2017.8393399.

[18] G. Singh, A. Kourmatzis, and A. R. Masri, "Volume measurement of atomizing fragments using image slicing," *Exp. Therm. Fluid Sci.*, vol. 115, Jul. 2020, doi:

10.1016/j.expthermflusci.2020.110102.

[19] J. Gao, W. Tan, L. Ma, Y. Wang, and W. Tang, "MUSEFood: Multi-sensor-based food volume estimation on smartphones," in *Proceedings - 2019 IEEE SmartWorld, Ubiquitous Intelligence and Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Internet of People and Smart City Innovation, SmartWorld/UIC/ATC/SCALCOM/IOP/SCI 2019*, Aug. 2019, pp. 899–906, doi: 10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00182.

[20] S. Uluisik, F. Yildiz, and A. T. Ozdemir, "Image processing based machine vision system for tomato volume estimation," in *2018 Electric Electronics, Computer Science, Biomedical Engineerings' Meeting, EBBT 2018*, 2018, pp. 1–4, doi: 10.1109/EBBT.2018.8391460.

[21] T. Y. Wang and S. K. Nguang, "Low cost sensor for volume and surface area computation of axi-symmetric agricultural products," *J. Food Eng.*, vol. 79, no. 3, pp. 870–877, 2007, doi: 10.1016/j.jfoodeng.2006.01.084.

[22] Google Android, "Android Studio Features | Android Studio," *Android*. 2016.

[23] OpenCv, "OpenCV Library," *OpenCV https://opencv.org/*, 2014.

[24] A. V. Kruglov, "Development of the rounded objects automatic detection method for the log deck volume measurement," in *First International Workshop on Pattern Recognition*, 2016, vol. 10011, doi: 10.1117/12.2242172.

[25] Sumit Saha, "A comprehensive guide to convolutional neural networks," *Medium*, vol. 1, no. 1, 2018.

[26] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2015, vol. 9351, doi: 10.1007/978-3-319-24574-4_28.

[27] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv*. 2017.

[28] S. Minaee, Y. Y. Boykov, F. Porikli, A. J. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image Segmentation Using Deep Learning: A Survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2021, doi: 10.1109/TPAMI.2021.3059968.

[29] S. Sural, G. Qian, and S. Pramanik, "Segmentation and histogram generation using the HSV color space for image retrieval," in *IEEE International Conference on Image Processing*, 2002, vol. 2, doi: 10.1109/icip.2002.1040019.

[30] C. Herbon, K. D. Tonnies, B. Otte, and B. Stock, "Mobile 3D wood pile surveying," 2015, doi: 10.1109/MVA.2015.7153101.

APPENDICES

Application Code

```
public class MainActivity extends AppCompatActivity implements
View.OnClickListener, UserInputDialog.UserInputDialogListener {


    private int REQUEST_CODE_PERMISSIONS = 101;
    private int CAMERA_PIC_REQUEST = 99;
    private final String[] REQUIRED_PERMISSIONS = new
String[]{"android.permission.CAMERA",
"android.permission.WRITE_EXTERNAL_STORAGE"};
    TextureView textureView_1;
    public static Bitmap PHOTO = null;
    ImageView bitmapIv;
    LinearLayout layout_b;
    private TextView
textView,textView1,textView2,textView3,textView4,textView5,
textView6,textView7, textView8,textView9,textView10,textView11,textView12,
            textView13,textView14,textView15,textView16,textView17;
    private Bitmap capturedBitmap;
    private Bitmap cannyBitmap;
    private Bitmap sobelBitmap;
    private Bitmap contourBitmap;


    ImageCapture imageCapture;
    ImageAnalysis imageAnalysis;
    Preview preview;
    String timbArea;
    String contourArea0;

    FloatingActionButton captureBtn, btnOk, btnCancel;

    static {
        if (!OpenCVLoader.initDebug())
            Log.d("ERROR", "Unable to load OpenCV");
        else
            Log.d("SUCCESS", "OpenCV loaded");
    }


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.find_volume_1);


        captureBtn = findViewById(R.id.btnCapture);
        btnOk = findViewById(R.id.acceptBtn);
        btnCancel = findViewById(R.id.rejectBtn);
        textView = (TextView) findViewById(R.id.kv_pixels);
        textView1 = (TextView) findViewById(R.id.timber_pixels);
        textView2 = (TextView) findViewById(R.id.kv_surface_area);
        textView3 = (TextView) findViewById(R.id.timber_surface_area);
        textView4 = (TextView) findViewById(R.id.timber_volume);
        textView5 = (TextView) findViewById(R.id.t_view);
```

```
        textView6 = (TextView) findViewById(R.id.t1_view);
        textView7 = (TextView) findViewById(R.id.t2_view);
        textView8 = (TextView) findViewById(R.id.t3_view);
        textView9 = (TextView) findViewById(R.id.t4_view);
        textView10 = (TextView) findViewById(R.id.log_length);
        textView11 = (TextView) findViewById(R.id.t5_view);
        textView12 = (TextView) findViewById(R.id.t6_view);
        textView13 = (TextView) findViewById(R.id.t7_view);
        textView14 = (TextView) findViewById(R.id.t8_view);
        textView15 = (TextView) findViewById(R.id.t9_view);
        textView16 = (TextView) findViewById(R.id.t10_view);
        textView17 = (TextView) findViewById(R.id.t11_view);

        btnOk.setOnClickListener(this);
        btnCancel.setOnClickListener(this);

        layout_b = findViewById(R.id.llBottom);
        textureView_1 = findViewById(R.id.textureView);
        bitmapIv = findViewById(R.id.ivBitmap);

        if (allPermissionsGranted()) {
            startCamera();
        } else {
            ActivityCompat.requestPermissions(this, REQUIRED_PERMISSIONS,
REQUEST_CODE_PERMISSIONS);
        }
    }

    private void startCamera() {

        CameraX.unbindAll();
        preview = setPreview();
        imageCapture = setImageCapture();
        imageAnalysis = setImageAnalysis();

        CameraX.bindToLifecycle(this, preview, imageCapture, imageAnalysis);
    }


    private Preview setPreview() {

        Rational aspectRatio = new Rational(textureView_1.getWidth(),
textureView_1.getHeight());
        Size screen = new Size(textureView_1.getWidth(),
textureView_1.getHeight()); //size of the screen


        PreviewConfig pConfig = new
PreviewConfig.Builder().setTargetAspectRatio(aspectRatio).setTargetResolution(s
creen).build();
        Preview preview = new Preview(pConfig);

        preview.setOnPreviewOutputUpdateListener(
                new Preview.OnPreviewOutputUpdateListener() {
                    @Override
                    public void onUpdated(Preview.PreviewOutput output) {
                        ViewGroup parent = (ViewGroup)
textureView_1.getParent();
                        parent.removeView(textureView_1);
```

```
                        parent.addView(textureView_1, 0);


textureView_1.setSurfaceTexture(output.getSurfaceTexture());
                        transform();
                    }
                });

        return preview;
    }


    private ImageCapture setImageCapture() {
        ImageCaptureConfig imageCaptureConfig = new
ImageCaptureConfig.Builder().setCaptureMode(ImageCapture.CaptureMode.MIN_LATENC
Y)

.setTargetRotation(getWindowManager().getDefaultDisplay().getRotation()).build(
);
        final ImageCapture imgCapture = new ImageCapture(imageCaptureConfig);

        captureBtn.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {


                imgCapture.takePicture(new
ImageCapture.OnImageCapturedListener() {
                    @Override
                    public void onCaptureSuccess(ImageProxy image, int
rotationDegrees) {
                        Bitmap bitmap = textureView_1.getBitmap();
                        capturedBitmap = bitmap;
                        cannyBitmap = bitmap;
                        sobelBitmap = bitmap;
                        contourBitmap = bitmap;
                        showAcceptedRejectedButton(true);
                        bitmapIv.setImageBitmap(bitmap);
                    }

                    @Override
                    public void onError(ImageCapture.UseCaseError useCaseError,
String message, @Nullable Throwable cause) {
                        super.onError(useCaseError, message, cause);
                    }
                });


            }
        });

        return imgCapture;
    }


    private ImageAnalysis setImageAnalysis() {

        HandlerThread analyzerThread = new HandlerThread("OpenCVAnalysis");
```

```
        analyzerThread.start();


        ImageAnalysisConfig imageAnalysisConfig = new
ImageAnalysisConfig.Builder()

.setImageReaderMode(ImageAnalysis.ImageReaderMode.ACQUIRE_LATEST_IMAGE)
                .setCallbackHandler(new Handler(analyzerThread.getLooper()))
                .setImageQueueDepth(1).build();

        ImageAnalysis imageAnalysis = new ImageAnalysis(imageAnalysisConfig);

        imageAnalysis.setAnalyzer(
                new ImageAnalysis.Analyzer() {
                    @Override
                    public void analyze(ImageProxy image, int rotationDegrees)
{

                        final Bitmap bitmap = textureView_1.getBitmap();


                        if (bitmap == null)
                            return;

                        Mat mat = new Mat();
                        Utils.bitmapToMat(bitmap, mat);



                        Utils.matToBitmap(mat, bitmap);
                        runOnUiThread(new Runnable() {
                            @Override
                            public void run() {
                                bitmapIv.setImageBitmap(bitmap);
                            }
                        });

                    }
                });


        return imageAnalysis;

    }

    private void showAcceptedRejectedButton(boolean acceptedRejected) {
        if (acceptedRejected) {
            CameraX.unbind(preview, imageAnalysis);
            layout_b.setVisibility(View.VISIBLE);
            captureBtn.hide();
            textureView_1.setVisibility(View.GONE);
        } else {
            captureBtn.show();
            layout_b.setVisibility(View.GONE);
            textureView_1.setVisibility(View.VISIBLE);
        }
    }
```

```
    private void transform() {
        Matrix mx = new Matrix();
        float w = textureView_1.getMeasuredWidth();
        float h = textureView_1.getMeasuredHeight();

        float cX = w / 2f;
        float cY = h / 2f;

        int rotationDgr;
        int rotation = (int) textureView_1.getRotation();

        switch (rotation) {
            case Surface.ROTATION_0:
                rotationDgr = 0;
                break;
            case Surface.ROTATION_90:
                rotationDgr = 90;
                break;
            case Surface.ROTATION_180:
                rotationDgr = 180;
                break;
            case Surface.ROTATION_270:
                rotationDgr = 270;
                break;
            default:
                return;
        }

        mx.postRotate((float) rotationDgr, cX, cY);
        textureView_1.setTransform(mx);
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {

        if (requestCode == REQUEST_CODE_PERMISSIONS) {
            if (allPermissionsGranted()) {
                startCamera();
            } else {
                Toast.makeText(this, "Permissions not granted by the user.",
Toast.LENGTH_SHORT).show();
                finish();
            }
        }
    }

    private boolean allPermissionsGranted() {

        for (String permission : REQUIRED_PERMISSIONS) {
            if (ContextCompat.checkSelfPermission(this, permission) !=
PackageManager.PERMISSION_GRANTED) {
                return false;
            }
        }
        return true;
    }

    @Override
```

```
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.rejectBtn:
                showAcceptedRejectedButton(false);
                break;

            case R.id.acceptBtn:
                File file = new File(

Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES),
"" + System.currentTimeMillis() + ".jpg");
                imageCapture.takePicture(file, new
ImageCapture.OnImageSavedListener() {


                    @Override
                    public void onImageSaved(@NonNull File file) {
                        showAcceptedRejectedButton(false);

                        Toast.makeText(getApplicationContext(), "Please enter
the values", Toast.LENGTH_LONG).show();
                    }

                    @Override
                    public void onError(@NonNull ImageCapture.UseCaseError
useCaseError, @NonNull String message, @Nullable Throwable cause) {

                    }
                });

                findVolume();

        }
    }

    public void findVolume () {
        Mat knownVolumeMask = new Mat();
        Mat originalMat = new Mat();
        Mat hsvMat = new Mat();
        Bitmap src = cannyBitmap;
        Mat knownVolumeCanny = new Mat();
        Mat timberCanny = new Mat();
        Mat hierarchy = new Mat();
        Mat hierarchy1 = new Mat();
        Mat timberMask = new Mat();


        Utils.bitmapToMat(capturedBitmap, originalMat);
        Imgproc.cvtColor(originalMat, hsvMat, Imgproc.COLOR_RGB2HSV);
        Scalar knownVolumelowerRange = new Scalar(50, 91, 54);
        Scalar knownVolumeupperRange = new Scalar(116, 255, 174);
        Scalar timberlowerRange = new Scalar(0, 113, 80);
        Scalar timberupperRange = new Scalar(95, 255, 255);
        Core.inRange(hsvMat, knownVolumelowerRange, knownVolumeupperRange,
knownVolumeMask);
        Core.inRange(hsvMat, timberlowerRange, timberupperRange, timberMask);

        List<MatOfPoint> contourList = new ArrayList<MatOfPoint>();
        List<MatOfPoint> contourList1 = new ArrayList<MatOfPoint>();
```

```
        Mat kernel = Imgproc.getStructuringElement(Imgproc.MORPH_OPEN, new
org.opencv.core.Size(2, 2));

        Imgproc.dilate(knownVolumeMask, knownVolumeCanny, kernel);
        Imgproc.dilate(timberMask, timberCanny, kernel);

        Imgproc.findContours(knownVolumeCanny, contourList, hierarchy,
Imgproc.RETR_EXTERNAL, Imgproc.CHAIN_APPROX_SIMPLE);
        Imgproc.findContours(timberCanny, contourList1, hierarchy1,
Imgproc.RETR_EXTERNAL, Imgproc.CHAIN_APPROX_SIMPLE);

        Mat kvcontours = Mat.zeros(knownVolumeCanny.size(), CvType.CV_8UC3);
        Mat tcontours = Mat.zeros(knownVolumeCanny.size(), CvType.CV_8UC3);
        Mat addcontours = Mat.zeros(knownVolumeCanny.size(), CvType.CV_8UC3);


        StringBuilder kvAreaList = new StringBuilder();
        for (int i = 0; i < contourList.size(); i++) {

            double cont_area = Imgproc.contourArea(contourList.get(i));
            if (cont_area > 500) {
                Imgproc.drawContours(kvcontours, contourList, i, new Scalar(0,
255, 0), -1);
                kvAreaList.append(cont_area);

                double cont_area0 = Imgproc.contourArea(contourList.get(0));
                contourArea0 = Double.toString(cont_area0);
                textView5.setText(contourArea0);
            }
        }
        StringBuilder logAreaList = new StringBuilder();
            for (int j = 0; j < contourList1.size(); j++) {

            double cont_area1 = Imgproc.contourArea(contourList1.get(j));
            if (cont_area1 > 500) {
                Imgproc.drawContours(tcontours, contourList1, j, new Scalar(0,
0, 255), -1);
                logAreaList.append(cont_area1);


            }
        }


        Core.bitwise_or(kvcontours,tcontours,addcontours);
        Imgproc.cvtColor(addcontours, addcontours, Imgproc.COLOR_RGB2RGBA);
        Imgproc.cvtColor(originalMat, originalMat, Imgproc.COLOR_RGB2RGBA);
        int timbpix = Core.countNonZero(timberCanny);
        timbArea = Integer.toString(timbpix);
        textView6.setText(timbArea);

        textView.setText("Known Surface Pixels");
        textView1.setText("Timber Pixels");
        textView12.setText(" pixels");
        textView13.setText(" pixels");
```

```
        Core.addWeighted(addcontours,0.5,originalMat,1,0,originalMat);

        openDialog();


        Utils.matToBitmap(originalMat, src);
        bitmapIv.setImageBitmap(src);
    }

    private void openDialog() {
        UserInputDialog userInputDialog = new UserInputDialog();
        userInputDialog.show(getSupportFragmentManager(), "User Inputs");
    }

    @Override
    public void applyTexts(String kvSurface, String log_Length) {
        textView7.setText(kvSurface);
        textView11.setText(log_Length);
        textView10.setText("Log Length");
        double param1 = Double.parseDouble(contourArea0);
        double param2 = Double.parseDouble(kvSurface);
        double param3 = Double.parseDouble(timbArea);
        double ratio = ((param3 / param1)*param2);
        double roundOffArea = (double) Math.round(ratio * 100) / 100;
        textView8.setText(Double.toString(roundOffArea));
        double length = Double.parseDouble(log_Length);
        double theVolume = ratio*length;
        double roundOffVolume = (double) Math.round(theVolume * 100) / 100;
        textView9.setText(Double.toString(roundOffVolume));
        textView2.setText("Known Surface Area");
        textView3.setText("Timber Surface Area");
        textView4.setText("Total Volume");
        textView14.setText(Html.fromHtml(" m<sup>2</sup>"));
        textView15.setText(Html.fromHtml(" m<sup>2</sup>"));
        textView17.setText(Html.fromHtml(" m<sup>3</sup>"));
        textView16.setText(" meters");

    }
}
```

Segmentation Tool Code

```
import cv2
import numpy as np

def trackChanged(x):
  pass


cv2.namedWindow('Trackbars')

cv2.createTrackbar("Lower - H", "Trackbars", 0, 179, trackChanged)
cv2.createTrackbar("Lower - S", "Trackbars", 0, 255, trackChanged)
cv2.createTrackbar("Lower - V", "Trackbars", 0, 255, trackChanged)
cv2.createTrackbar("Upper - H", "Trackbars", 179, 179, trackChanged)
cv2.createTrackbar("Upper - S", "Trackbars", 255, 255, trackChanged)
cv2.createTrackbar("Upper - V", "Trackbars", 255, 255, trackChanged)


img = cv2.imread('C:/Users/ziqbal/anaconda3/envs/image2.jpg')
img = cv2.resize(img, (0,0), fx=0.8, fy=0.8)
imgHsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

while(True):
   low_h = cv2.getTrackbarPos("Lower - H", "Trackbars")
   low_s = cv2.getTrackbarPos("Lower - S", "Trackbars")
   low_v = cv2.getTrackbarPos("Lower - V", "Trackbars")
   up_h = cv2.getTrackbarPos("Upper - H", "Trackbars")
   up_s = cv2.getTrackbarPos("Upper - S", "Trackbars")
   up_v = cv2.getTrackbarPos("Upper - V", "Trackbars")

   lower_range = np.array([low_h, low_s, low_v])
   upper_range = np.array([up_h, up_s, up_v])
   firstMask = cv2.inRange(imgHsv, lower_range, upper_range)
   result = cv2.bitwise_and(img, img, mask=firstMask)
   secondMask = cv2.cvtColor(firstMask, cv2.COLOR_GRAY2BGR)

   cv2.imshow("imgHsv",imgHsv)
   cv2.imshow("firstMask",firstMask)
   cv2.imshow("result",result)
   cv2.imshow("secondMask",secondMask)
   stacked = np.hstack((secondMask,img,result))
   cv2.imshow('Trackbars',cv2.resize(stacked,None,fx=0.5,fy=0.5))

   if cv2.waitKey(1) == 27:

       break

cv2.destroyAllWindows()
```

LIST OF FIGURES