University of Nottingham
UK | CHINA | MALAYSIA

Department of Mechanical, Materials and Manufacturing Engineering

Improving Nutritional Tracking: Combining Close-Range Photogrammetry and Machine Learning

MMME3083 INDIVIDUAL BEng PROJECT FINAL DISSERTATION 2024-25

Author: Noah Santaub

Student ID: 20345518

Date of report: 13th May 2025

Supervisor: Prof. Samanta Piano

Summary

This report summarises a bachelor's individual project to develop a pipeline of close-range photogrammetry and food recognition, to calculate the calories in an item of food. Photogrammetry has been used in many fields, such as manufacturing, agriculture and archaeology. With support from previous studies photogrammetry can be repurposed to provide an alternative method of tracking nutritional food consumption. Current methods are found to be inconvenient or impractical but the proposed method of taking global pictures of the food is found to be more practical and accurate than using a scale or AI. Tracking calories and other food intake is of high importance to several groups of people, from those who are athletes and eating higher calories than average to help sustain their energy requirements or possibly improve their performance, to those recovering from chemotherapy who must meet targets of nutritional intake to aid in their recovery.

A pipeline has been created that demonstrates how photogrammetry and food recognition can be connected to estimate the calories of fruit. Laboratory testing was conducted to verify the pipeline could provide accurate nutritional information from a dense point cloud, leading to a finding of 21% MAPE thus validating the pipeline. When compared against existing apps, the pipeline was found to be more accurate with a delta of 9% MAPE for apps that require a scale and 17% MAPE when using one with AI. Closing the gap between the pipeline and applications requires automation, from point cloud generation to calorie calculation, as the current method is quite labour intensive.

Mobile implementation of the pipeline is reliant on calibration. This is the most important factor that influences the accuracy of nutritional values, thus multiple methods were investigated, and a printed checkerboard was chosen. While it has a larger error compared to lab testing, with 52% MAPE, the pipeline is still supported but restricted by calibration, hence requiring further research.

Multiple food recognition methods have been reviewed and lead to the finding of the object detection algorithm YOLO11n being most appropriate; with its enhanced accuracy over other object detection algorithms and reduced complexity of integrating bounding boxes into an image recognition model. Investment is required to create an appropriate food dataset to ensure high accuracy and reliable detection.

Nomenclature

Symbol	Quantity	Unit
kcal	Calorie	kJ
m	Meter	m
v	Volume	m^3
ρ	Density	kgm^{-3}

Acronyms

Acronym	Meaning
3D	Three Dimension
AI	Artificial Intelligence
API	Application Programming Interface
CNN	Convolutional Neutral Network
COCO	Common Objects in Context
CPU	Central Processing Unit
Faster R-CNN	Faster Region Convolutional Neural Network
GPU	Graphics Processing Unit
LVIS	Large Vocabulary Instance Segmentation
MAPE	Mean Absolute Percentage Error
ML	Machine Learning
NN	Neutral Network
OD	Object Detection
RAM	Random Access Memory
VRAM	Video Random Access Memory
YOLO	You Only Look Once

1. Introduction

1.1. Background and Context

From the beginning of civilisation food has been the centre on which everything revolves. Fire was discovered, leading to a new method of preparing food, and in some foods cooking increases calories and nutritional values [1]. As humanity evolved the relationship with food changed, from merely eating for survival, to a social occasion [2].

Calories are a measure of the amount of energy in foodstuffs, units of kcal (kilocalories). The recommended average daily calorie intake is 2500 kcal for men and 2000 kcal for women [3]. However, calorie and nutritional intake is dependent on individual needs, whether physically active, in recovery or dieting. Exerting energy requires energy, so when being physically active it is important to balance energy intake and output, while also considering recovery. With 90 minutes of exercise per day, athletes are recommended to consume around 50 kcal/kg/day. Tracking their food intake and calorie consumption can aid in performance but also identify flaws in a diet; especially if they are nutrient deficient or too nutrient rich and reduce the need for supplements such as protein supplements or vitamins [4]. For those with cancer it is vital to prevent malnutrition, as studies show the importance of ensuring the body meets its energy requirements, otherwise this can have an adverse effect on recovery and how they respond to treatment [5]. Further, for those undergoing chemotherapy it is advised to consume between 1 to 1.5 grams of protein/kg/day [6], this is highly difficult to achieve without tracking nutritional consumption. While there are clear benefits of tracking and counting calories and other values, it can foster an unhealthy relationship with food, resulting in developing an eating disorder [7]. Thus, it is not recommended that everyone should track the calories they consume however, it is a useful tool for those who have a relatively healthy relationship with food.

With the rise of technology, a method of tracking the total calories consumed and burnt throughout the day has become easier than ever. Smartphones and smartwatches can track your steps to give you an estimation of the number of calories burnt. There are numerous applications for your smartphone that help you track calories; with most of them you enter the food type and the weight, and it gives you a list of calories and additional nutritional information [8-10]. This is highly impractical when the user is commuting and does not have access to a scale to enter the weight. Furthermore, people tend to overestimate the number of calories in food [11], thus this would introduce errors in the calculated calories consumed. Other more sophisticated apps allow the user to take an image of the food and return the nutritional information by utilising artificial intelligence, AI [12]. However, this method of calculating the size of the food has its limitations in accuracy, also contributing to the inaccuracy in calculating calories.

In this project, close-range photogrammetry will be employed to obtain dimensional measurements of food along with food recognition, as the food will be automatically identified and hence calculate the calories and nutritional values. The hypothesis is such that the method will result in higher accuracy of calorie estimation than current software.

1.2. Similar Studies

There are few existing research papers on generating a point cloud, by taking many images of food in a circular path, to calculate volume and or its calories. Although, there are a few studies with similar aims that use different techniques to the one proposed in this project.

Kong et al [13] proposed a solution of software that provides a dietary assessment however, it does not utilise photogrammetry software to create a point cloud and find the foods volume. Instead, it simplifies the volume of an object as it models an apple as a sphere instead of its individual geometry. Whereas “for an arbitrary shaped object” a tetrahedral mesh is created with the volume calculated via a geometric equation. Assuming foods can be modelled with perfect geometry can lead to inaccurate calorie values.

Many smartphones have multiple cameras and Ando et al [14] created a mobile app that takes advantage of multiple cameras to use it as a stereo camera system. This allows for the depth value to be calculated and reduces the need for further extrinsic camera calibration. This produces a small mean error but a relatively large standard deviation error in estimated calories. Rahman et al [15] also used a stereo camera system for volume estimation resulting in a small error.

The most similar studies to the proposed method are by Puri et al [16] and Dehais et al [17]. With a similar methodology in food volume estimation, they created an incomplete point cloud, and the methods use three and two images respectively to create the point cloud. This is drastically less than what is normally considered to generate an accurate reconstruction. While both have a low error rate, more images will lead to a more accurate point cloud and thus a lower error rate.

Instead of the conventional methods of calibration (discussed in section 2.1), Yue et al [18] used specific containers with known dimensions to calibrate and scale images. Length and thickness of the food was selected manually, and this is very inefficient if developed into a smartphone application.

1.3. Aim and Objectives

The aim of this project is to develop a pipeline that calculates the calories of food using close-range photogrammetry and food recognition via machine learning.

Objectives

1. Explore research and existing apps that calculate calories and identify their flaws to assess the use of photogrammetry and food recognition machine learning in this project.
2. Identify an appropriate machine learning model for food recognition, conduct further training and validate said model in identifying food in an image.
3. Create a point cloud using photogrammetry software to accurately calculate the volume of the food, along with its density, so that the calorie value can be found.
4. Validate the proposed pipeline and its nutritional accuracy by comparing the experimental data to existing software.

2. Literature Review

2.1. Photogrammetry

Defined by Luthmann [19] photogrammetry is an overarching term of the method for image measurement and interpretations to derive the shape and location of an object from one or more photographs of said object. Photogrammetry is used in many sectors such as automotive, aerospace, agriculture and surveying to name but a few. It is most used over other optical or contact metrology techniques when those methods are incapable of retrieving sufficient geometric information from the object. Close range photogrammetry is commonly defined with the subject location being under 200m. Close range photogrammetry requires multiple components to deliver successful results, calibration (parameters and targeting), image capture (image configuration and processing) and reconstruction (3-dimensional (3D) reconstruction and interfaces) [20].

Camera calibration is used to create a virtual camera model described by intrinsic parameters, centre of images' location in a coordinate system and error parameters such as distortion. By accounting for the intrinsic parameters, a pinhole camera model can be assumed and hence with the use of the generated coordinate system the image can be transformed to represent the 'true' size of the object captured. The two common variants of calibration are test-field calibration and self-calibration [21].

Test-field calibration can be conducted before or after conducting the photogrammetry measurements using a suitable target of identifiable points with known distances. Most commonly this is conducted using a checkerboard type object, a grid of black and white squares, as seen in Figure 1. Photos are taken at different orientations to ensure suitable information was obtained. The extrinsic parameters are calculated via the object coordinates of the test field. Intrinsic and extrinsic parameters are bundled and with known test field coordinates a scale can be derived and used when the system is deployed. This method is done when other calibration methods cannot be performed or to aid in the evaluation of a camera. The test field should be representative of the actual object to be measured and due to the nature of cameras, changing the focal length of lens would change the previously calculated distortion value. Thus, no changes can be made from conducting the test field calibration to object reconstruction or vice versa [20].

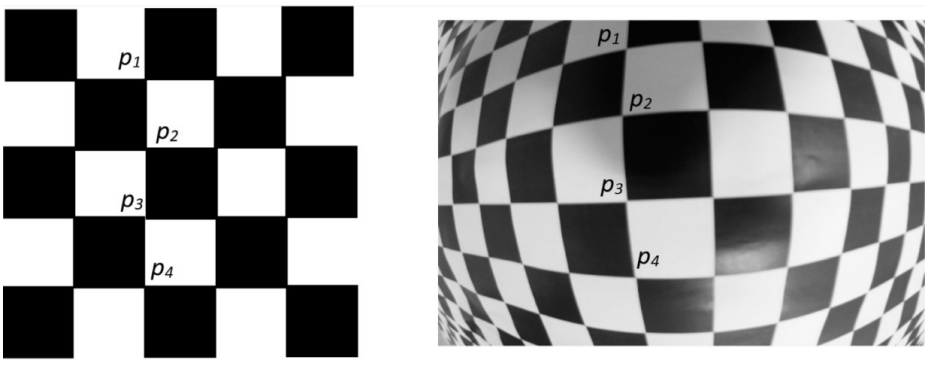


Figure 1 Left checkerboard image with identified reference points. Right distorted image captured from a camera [20].

Self-calibration, as its name implies, uses the object itself for calibration. The cameras' intrinsic parameters are calculated using the captured images and provides high accuracy in reconstruction. Reference points, commonly termed as targets, are used to generate a global coordinate system that defines extrinsic parameters. Additionally, these targets provide a scale to the coordinate space [19].

Camera specifications can directly impact the quality of the point cloud. It was found that having a large pixel size with a smaller sensor area has greater accuracy [22]. Further, a fixed lens is more appropriate for close range photogrammetry, as it was found by Shortis et al [23] that there are instabilities and challenges when

calibrating a camera with a zoom lens. Dedicated cameras such as DSLRs are more appropriate in some cases to smartphones, although the environment and operator can also impact the effectiveness of the camera used [24].

Photogrammetry software is required for image processing and 3D reconstruction [20]. There is a variety of commercial and open-source software that generate a point cloud from images. Agisoft Metashape is a commercial photogrammetry package developed by Agisoft, a Russian company, [25]. It has become the standard for photogrammetry processing and reconstruction, due to its high point cloud density, percentage of model completion and low percentage of noise generated [26]. Open-source alternatives which are commonly free is a benefit but often provide less accurate results than commercial options, particularly Agisoft Metashape. Users can see how the program functions compared to the black box nature of propriety software, and community development allows for continuous improvement reducing the accuracy gap between open-source and commercial and development of new features [27, 28].

2.2. Food Recognition

Creating a program that can detect and adapt to changes that even the designer could not identify was impossible, thus a new technology with the ability to learn and adapt was developed, machine learning, ML. Computational models of the brain were made to simulate and study how we learn, from this came neural networks [29].

With research in neural networks, deep learning was becoming a focus of research. Deep learning is the method of a self-learning machine that utilises multilayer models and vast datasets [30]. By linking multiple simple models together each layer transforms the data, leading to a high-level representation. One variant of deep learning is a convolution neural network (CNN). CNNs, seen in Figure 2, have an input layer which passes to the convolution layer where different features are detected. Pooling layers calculate average feature maps. Further convolutional and pooling layers are used to increase the resolution of the data observed by increasing the number of iterations while having a smaller dataset to work from. Finally, it returns the maximum value output from the “neurons”, commonly referred to as the predicted value from the model [31].

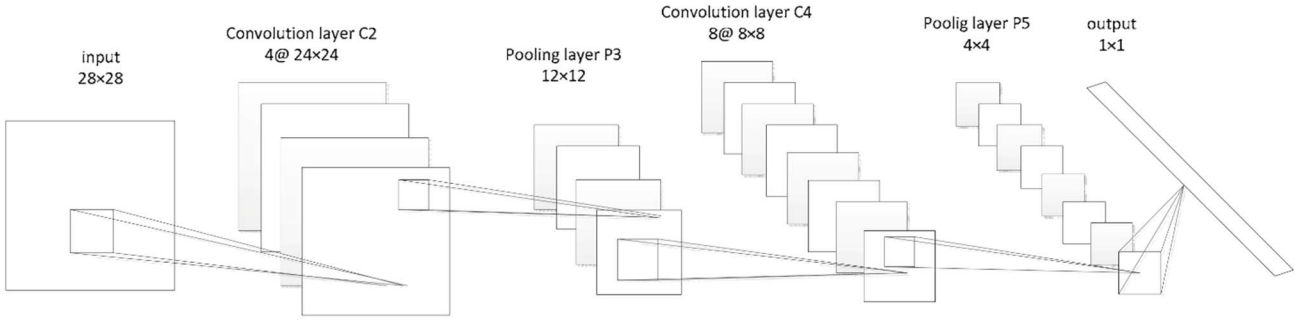


Figure 2 A simple structure of a CNN [31]

CNNs are used in food recognition and are the preferred method of image classification due to the model structure and training methods [32].

The data on which a ML model learns is of key importance, as its quality directly impacts performance. While there is no significant performance penalty for using a small dataset it is important that the data is representative of the use case and contains sufficient information, this can be achieved by distorting the images or adjusting image quality [33]. If a significantly large dataset is used, overfitting becomes a challenge, overfitting is where the model starts to predict incorrectly due to its reliance on the data it was trained with, commonly due to verifying the model with the same data it was trained with. To overcome overfitting, parameters need to be finely tuned along with appropriate feature selection [34].

3. Methodology

3.1. Pipeline

The developed pipeline in this project is derived of two major sections in accordance with the project, close-range photogrammetry and machine learning, as seen below in Figure 3. This pipeline has been designed so that with minimal adjustments it can be implemented into a functioning mobile application.

Close-range photogrammetry has a well-established and accepted pipeline [20] and is used to generate a point cloud, where the food's volume is estimated via a point cloud processing software, as explored in 3.2. Various methods of calibration were investigated to find the balance between accuracy and practicality, 3.2.2.2. Food recognition is used to automate the identification process of the food, to reduce the burden on the user of searching through a list of foods. In principle, it is a simple task, but as seen in 3.3 and 4.2 challenges arose. With the identified food, a database containing nutritional information is used to determine the nutrients of said food. From this volume and nutritional information, the food's calories are calculated, 3.4.

Fruit was chosen as the test food due to the simplicity of capturing images of solid objects, rather than full dishes, while also allowing for more repeatability and reducing the complexity of food detection as covered in 3.3. Five fruits were chosen based on size, ensuring they are not too big for image acquisition but large enough for a detailed reconstruction and availability. These were an apple, banana, orange, peach, and a pear.

All the software is run locally on a Windows 11 computer with the hardware: Intel Core i9-10885H CPU, NVIDIA GeForce RTX 2060 Max-Q GPU, 32 GB RAM and 2 terabytes secondary storage. A more powerful computer would likely deliver varying results as discussed in 4.1.1 and 4.2.2. Converting this project into an application would require servers, thus these results also provide a baseline of the technology required to improve both the results and the programs operation.

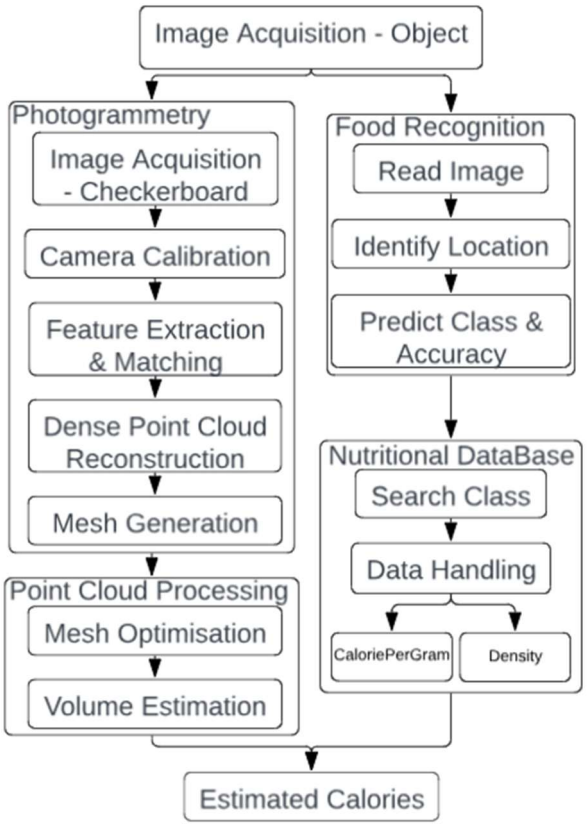


Figure 3 Proposed pipeline diagram

3.2. Photogrammetry

3.2.1. Laboratory Testing

Evaluation of the pipeline was first conducted in a laboratory environment, in which a proof of concept was developed to provide evidence warranting further research. Testing in such an environment reduced variability in data.

The apparatus can be seen in Figure 4. It consists of a Canon EOS 1300D with a Canon EF-S 18-55mm f/3.5-5.6 IS II lens and a rotating stage connected to a workstation. The camera is controlled remotely via a USB, using digiCamControl software [35]. This software allows for remote focus selection and fine tuning of intrinsic parameters such as shutter speed, colour temperature and saturation adjustment. Removing possible interferences such as an operator touching the camera's controls, while also allowing for easier operation.

First, images of the checkerboard were captured with the object being captured later. The order of operations here are independent and no camera parameters change between object and calibration image acquisition.

As seen in Figure 4, a checkerboard is placed at the centre of the stage and field of view, which is selected as the cameras focus point. A ceramic matte checkerboard, WMO-060-4.0-C, is used for calibration as it is most suitable for this experiment. Using a proprietary MATLAB script, created by the Nottingham Manufacturing Metrology Team [36], the rotating platform which is connected to a stepper motor rotates a defined number of steps and captures an image at that angle of rotation. In this instance each step was six degrees, resulting in 60 images of the target object. Once the script ended, the resulting images were passed through to the MATLAB image calibration app [37] and a further propriety script which generated a xml file, such that the aforementioned chosen photogrammetry software can receive the data, and generate a calibrated three-dimensional representation of the fruit, to be captured later.

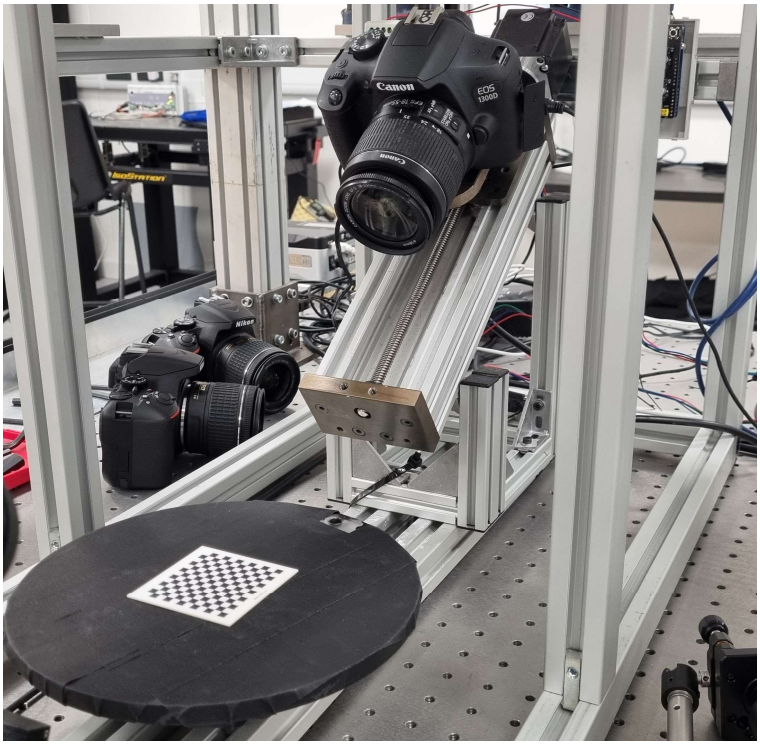


Figure 4 MMT Close Range Photogrammetry Apparatus with a ceramic checkerboard in the centre of the stage.

Additionally, the code generated a graphical representation of the camera plane in reference to the target object, in this case the checkerboard. Figure 5 shows the location at which each image was taken, in red, with the vertices of the checkerboard in blue.

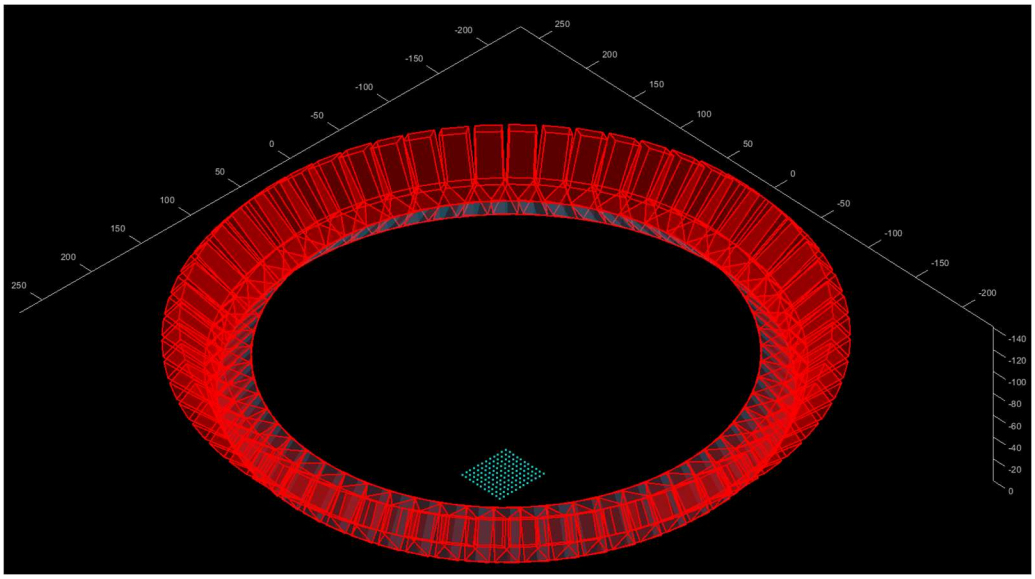


Figure 5 Camera position reconstruction during calibration.

Image acquisition stage consisted of replacing the checkerboard at the centre of the stage with one of five pieces of fruit and using the same image capture code as above, making sure the focus and other parameters do not change between each capture iteration.

From these image sets they were imported into Agisoft Metashape, along with the cameras' calibration file. After aligning the images, a point cloud was built to a high quality with densification and exported with the local geometry. Resulting in a generated point cloud in relation to the image plane, as shown in Figure 6. Screen captures of other fruit can be seen in the appendix. It was used to aid identifying whether something failed along the process and requires revisiting, as seen in 4.1.2.

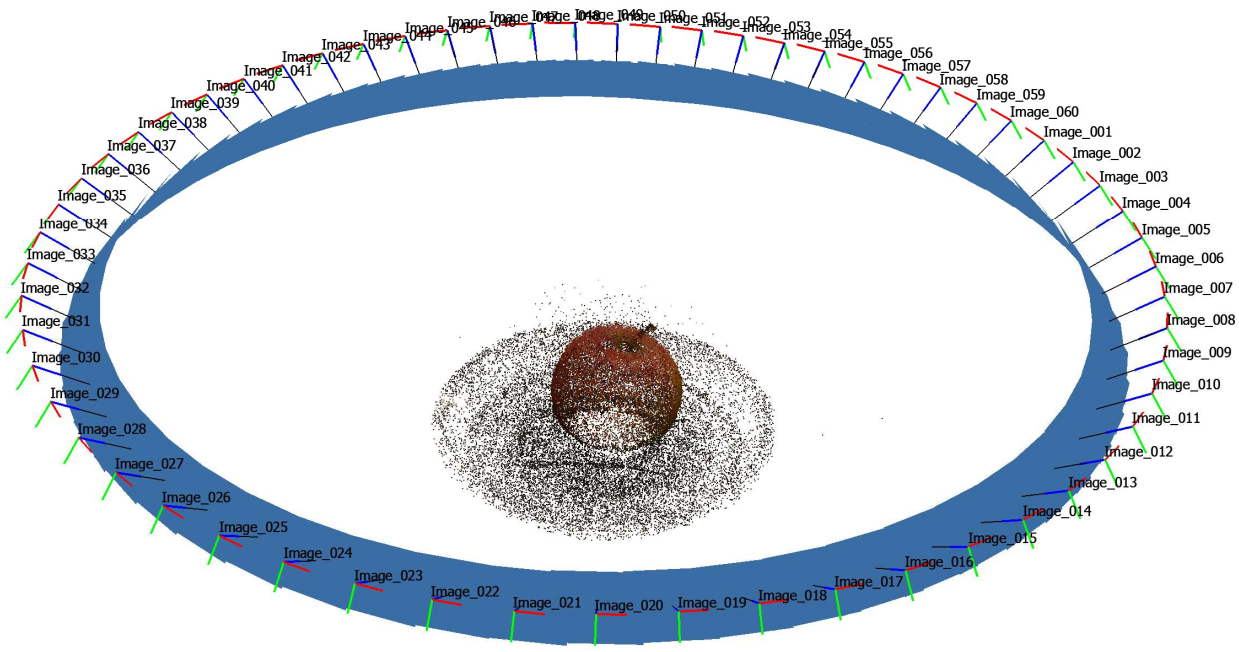


Figure 6 Agisoft screen capture of point cloud and camera poses.

As mentioned, in 3.4 the fruit's volume and mass are required to predict its calories. To do so requires a point cloud processing software, CloudCompare [38] was used, as it is free, having an integrated volume calculation function along with receiving training on it. CloudCompare contains a segment tool that allows the user to remove the unwanted and erroneous pixels, resulting in a cleaned point cloud, Figure 11. With this, the volume estimation tool, 2.5D volume, creates a depth map and using relative height derives a unitless volume, as seen in seen in 4.1.1.

3.2.2. Mobile Testing

As the lab testing resulted in a valid proof of concept, further testing was conducted. Where, instead of a professional camera and rotating stage, a smartphone and notebook were used, Figure 7. A Samsung S21+ 5G was used, with its primary camera being 12 MP, 1/1.76-inch sensor, f/1.8 [39].

In place of a controlled rotating stage a black hardcover notebook was used due to availability and colour. Having a neutral colour aids in feature recognition and provides a sharp contrast between the background and target object, simplifying the processes of cleaning a point cloud. While a plate could be used as a stage, having a concaved surface would prevent the use of conventional calibration methods as they must lay flat, thus a plate was not suitable for use in this approach. Although the method of image acquisition with a phone varies, the procedure of deriving a volume remains consistent with 3.2.1.

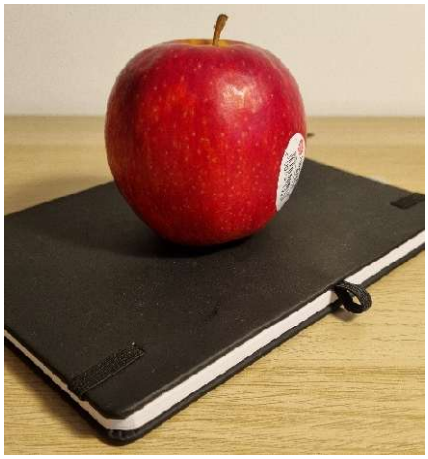


Figure 7 Mobile testing image acquisition of an apple on a black notebook.

3.2.2.1. Image Acquisition

To capture global images of the fruit to generate a successful point cloud, the camera must rotate circularly around the object. The initial approach consisted of holding the phone at the same position, rotating the object a few degrees and capturing the image each time. The second and accepted approach, as seen in Figure 8, was to use a 3D-printed phone stand (due to cost & accessibility) which held the phone in place while the target object was rotated, for greater feature recognition. With a confirmed method of image acquisition, the pipeline is followed as addressed above and the resulting data can be found in 4.1.2.1.

Figure 8 Mobile testing image acquisition setup, phone stand and staged a determined distance apart.

3.2.2.2. Mobile Calibration

With the nature of nutritional tracking occurring in various environments, each time an object needs to be measured calibration is required. With the goal of this project resulting in a free application, the calibration method must also be free or at minimum affordable. The checkerboard used in 3.2.1 costs £300+ due to its speciality [40]. Thus, various calibration methods were explored, printed checkerboard, digital checkerboard, printed coded scale bar and probe objects.

As established, checkerboard calibration is the most common and one of the most accurate calibration methods. With the checkerboard used in the lab, and similar types, being prohibitively expensive, a printed checkerboard was used to investigate its viability. A 4mm 10x17 grid checkerboard was generated via the camera calibration company calib.io [41] and laser printed on white card due to availability and accuracy. The checkerboard is placed on the same point on the notepad as the fruit, ensuring the focal length is consistent between acquisition stages, Figure 9.

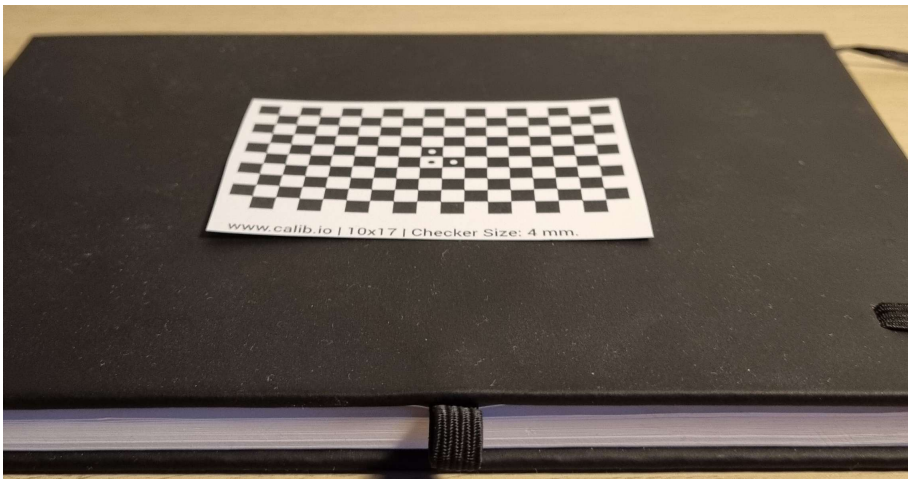
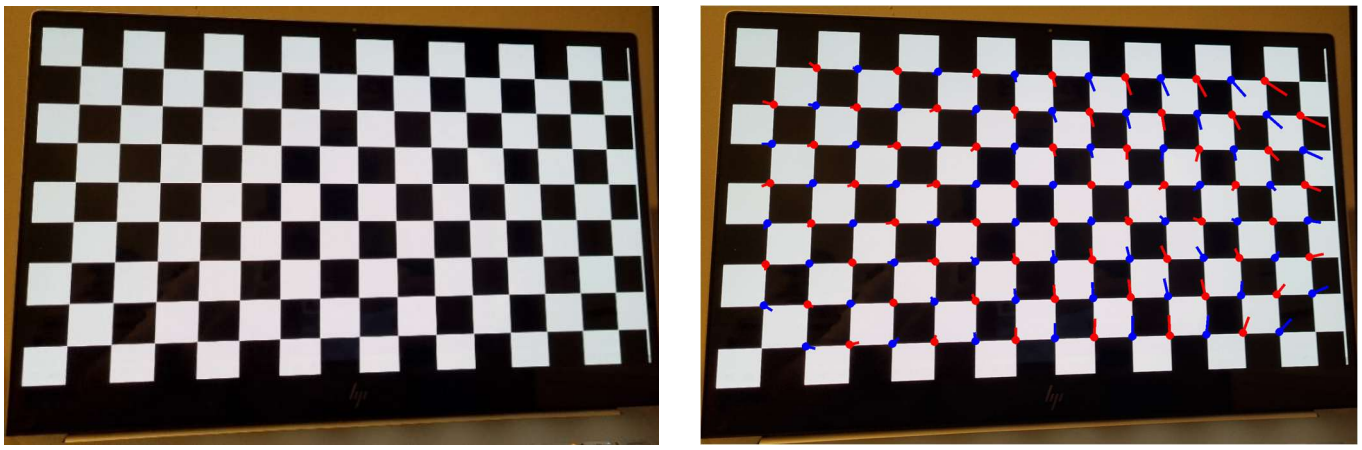


Figure 9 Sample calibration image of the 10x17 printed checkerboard

Agisoft Metashape has an automatic lens calibration tool which calculates the cameras parameters by capturing images of a checkerboard on a display [42]. Figure 10a is a sample image of what the digital checkerboard is. Figure 10b is a graphical representation of distortion, post camera calibration, with the lines representing the distance and direction between the captured pixel vertex and its known position.



a) b)

Figure 10 a) 15-inch OLED display photographed with the phone used. b) digital checkerboard with distortion markers.

Scale bars are a method of calibration whereby reference lengths allow the reconstruction to be scaled such that it represents the true size of the object, commonly used in archaeology and criminology. It is a highly convenient method as it only requires one round of image acquisition rather than two, as the scale bar must be placed by the object during the stage of the pipeline [20], seen in Figure 18. A printed scale bar with coded targets [43] was used similarly due to cost.

An alternate scaling calibration method could be achieved by probe objects; these are objects with known size being place near the target object during the image acquisition phase. After the point cloud has been generated the probe is measured on the point cloud processing software and the resultant delta against the true size is used to scale the point cloud; in this instance both a pound coin and a credit card are used as probes. A comparison between said calibration methods effect on accuracy can be found in 4.1.2.

3.3. Food Recognition

3.3.1. Image Recognition

Image recognition is the method of deep learning whereby a computer can identify the scene or object in an image [44]. There are multiple methods of implementing image recognition to the pipeline, such as an “off-the-shelf” solution, where an image recognition application exists which is pre-trained or can be trained to recognise food. Alternatively, creating a custom model can allow for greater control, or by splitting the difference and take an existing food recognition model and adapting it to meet the needs of this project.

Three models were selected to compare against each other to empirically conclude which model is the best for food recognition: Google Vision API [39], Google Vertex AI Vision [41] and a custom model. Google’s Vision API and Vertex AI Vision are used as free trials for the purpose of demonstrating the viability of the pipeline but at scale will introduce cost.

The simplest tool used was Google’s Vision API, due to no required training. An image is passed to the API and returns the predicted class with reasonable accuracy, as do all the models tested. Vertex AI Vision operates the same as above but allows for transfer learning. Transfer learning allows for a model to be further trained on a smaller more focused set of data, which in theory will increase the model’s accuracy.

A custom model was developed based on the published fruit recognition source code by Datalira [45] as a reference guide. The custom model was written in Python 3.9 using TensorFlow 2.10.1 as the AI framework, because of proficiency. Training the model consisted of selecting an appropriate existing architecture, DenseNet201, defining training parameters, and linking it to an image dataset. The code can be found in the appendix, Figure 28 and Figure 30 or on GitHub [46]. DenseNet201 was chosen as it outperformed other models after a single epoch, Figure 31. After training, the accuracy of the model on the test images from the dataset was 99.63%, Figure 29.

For the latter two models, a training dataset was required. The same dataset used by Datalira was selected as it contained over 40,000 images and 15 varieties of fruit [46]. It contained a sufficient number of images to ensure an accurate model, while also including noise to prevent model overfitting and poor learning.

The best performing model would then be implemented into the pipeline. However, in the case of when multiple objects were in a single image they would not be detected, increasing the inaccuracy and diminishing the benefits of ML implementation. It would require extensive computer vision integration to solve, but a solution was found, object detection.

3.3.2. Object Detection

Object detection, OD, is the method of deep learning, in which it identifies the locations and instances of each object in a frame [44]. This method solved the issue experienced above, as it automatically generates a bounding box while predicting the objects, allowing for multiple objects to be measured at one time. A bounding box is a frame encompassing an object of interest in the image.

Multiple object detection algorithms exist but the two major ones are YOLO [47], you only look once, and Faster R-CNN, faster region convolutional neural network [48]. YOLO was chosen over Faster R-CNN as it generally has higher accuracy and precision. Additionally, it is faster due to having a single-stage detection architecture compared to Faster R-CNN's two-stage detector with the intention of increasing accuracy [49]. YOLO was also selected due to its extensive documentation and community development. YOLO11 [50] was selected, over the highly researched version v8, because of increased accuracy and efficiency; with version 11n, nano, being most suited for transfer learning [51].

Two scripts were made, the first, Figure 32, loads in YOLO11n, trains it on the selected dataset and saves the trained model for future use. Second, Figure 33, takes one of the images from the photogrammetry image acquisition stage, passes the image though the model and sanitises the outputs class and accuracy, to identify what fruit variety to call from the nutritional database.

3.4. Nutritional Information

Once the food variety is identified the nutritional values need to be determined. There are several sources that provide such information: USDA FoodData Central Database [52], FooDB [53], and Nutritionix [54]. Each contains a vast amount of food types and in-depth nutritional values for each entry. Nutiritionix was selected due to having a larger amount of data, compared to USDA, and the API was free (for the projects scope) along with greater documentation, allowing for faster implementation. Although an API requires a constant internet connection, it doesn't require the user to store a large dataset locally on their device, unlike repositories.

However, in all available databases there is limited or zero data on density. This is an issue due to the nature of how calories will be calculated in the pipeline. Thus, a food density database was created, using existing studies to provide reputable values, Table 1.

The calories are calculated via Equation 1, it takes the mass of the object and a scaling constant to return the estimated calories of the measured fruit. The scaling constant here is called CaloriePerGram which uses the chosen database to find how many calories there are to a gram of the specific food type, Equation 2.

$$kcal_{Estimated} = v_{Point\ Cloud} * \rho_{food} * CaloriePerGram \quad \textit{Equation 1}$$

$$CaloriePerGram = \frac{kcal_{true}}{100gram} \quad \textit{Equation 2}$$

4. Results

4.1. Photogrammetry

4.1.1. Laboratory Testing

Following the procedure shall result in dense point clouds of the target objects. Figure 11 contains each cleaned point cloud from the lab testing. They require ‘cleaning’ to isolate not only the object pixels from the stage, but also the unwanted pixels to accurately determine the shape and thus its volume. Erroneous pixels can arise from changes in the image by the object moving or due to the reconstruction algorithm used.

There are voids in the point clouds which make them appear sparce however, this is a result of insufficient feature recognition which can be a result of overhangs, obfuscating the view of certain features, or there’s an insufficient number of images. The banana had significant artifacts because it did not fit in frame in all images, subsequently reducing the quality of feature matching that can be achieved and thus there were numerous sparse areas.

Figure 11 Cleaned point clouds of lab test fruit

2.5D volume calculation compares a constant source to the point cloud, allowing the generation of depth maps and thus volume, Figure 12. It uses interpolation to stitch the empty pixels of the point cloud. Step size has an inverse relationship with grid size & number of cells. With a higher number of cells, the resulting volume becomes more accurate. Geometry can impact the cell size limiting how refined it can be. However, more cells require more compute and is bottlenecked by the computers RAM size, the smallest step sizes were used where possible.

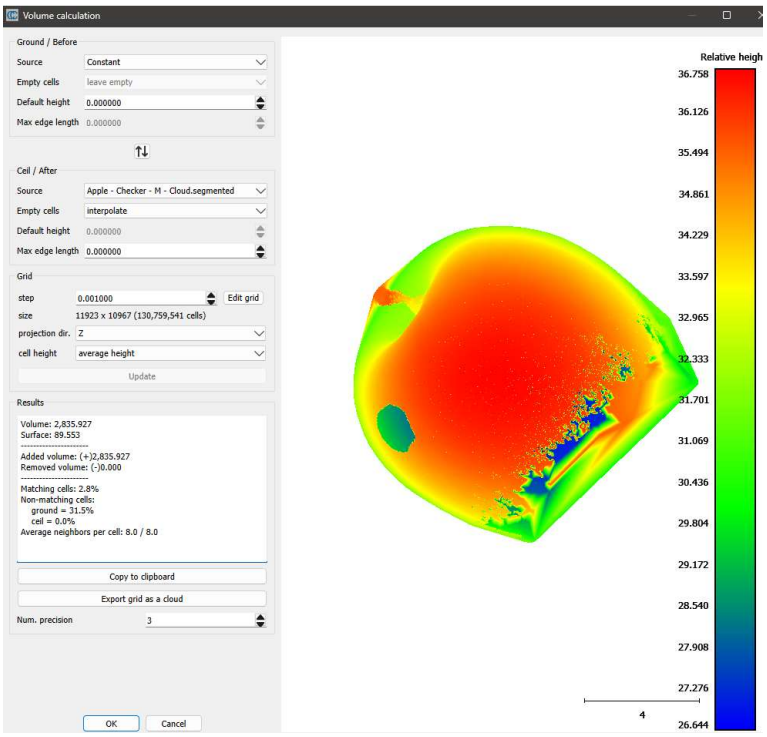


Figure 12 CloudCompare’s 2.5D volume calculation tool depth grid screen capture.

As seen in 3.4, mass is required to calculate calories, hence Equation 3 was derived which is based on calculating mass via density.

$$Estimated\ mass\ (kg) = \frac{Pixel\ volume}{Conversion\ factor} * density\ (kgm^{-3}) \qquad \textit{Equation 3}$$

The resulting volume from CloudCompare was in the same units as the formatting of the imported file into the software, thus the pixel volume is mm^3 as the calibration file used is in mm. Hense a conversion factor is used to transpose the pixel volume into SI units, m^3. In this instance the conversion factor $= 1e + 9$. Equation 4 contains an example of Equation 3 when using the data of the apple, with all values displayed in Table 1.

$$Estimated\ apple\ mass = \frac{126{,}694.701}{1\mathrm{e} + 9} * 950 = 0.12036\ kg \qquad \textit{Equation 4}$$

Fruit Type (Variety)	Pixel volume	Density, kgm^{-3}	Estimated mass, g
Apple (Pink Lady)	126,694.70	950 [55]	120.36
Banana (Colombian)	114,380.86	1100 [56]	125.81
Orange (Lane-late)	108,673.71	900 [57]	97.81
Peach (Spring Flame)	94,848.73	998 [58]	94.66
Pear (British Conference)	96,160.00	1036 [59]	99.62

Table 1 Estimated mass of fruit from lab data

From using Equation 3, the fruits' calories are calculated along with a percentage error of mass & calories to evaluate the accuracy of this method, Table 2. Calories and mass have the same percentage error due to calories being directly proportional to mass, as observed in Equation 1. Accuracy is considered good below 20 percent and reasonable between 20 to 50 percent. The overall percentage error is on the boundary with 21%. While slightly higher than existing research, it supports this theory and approach of calorie calculation and thus determined this method was needed to be further investigated. The high errors are a result of missing information due to shadows, overhangs and failed feature matching. The banana has a significant error due to the entire object not being in each image, resulting in poor feature recognition and thus a poor-quality point cloud, seen in Figure 11.

Fruit Type	True, g	Estimated, g	True, $kcal$	Estimated, $kcal$	Error, %
Apple	144	120.36	74.88	62.59	16.41
Banana	153	125.81	126.17	111.97	17.77
Orange	135	97.81	63.45	45.97	27.55
Peach	115	94.66	44.85	36.92	17.69
Pear	134	99.62	62.98	46.82	25.66
			Mean absolute percentage error		21.02

Table 2 Mass & calorie estimation percentage error

The nominal errors are slightly reduced when comparing calories, which causes the impression it is less accurate than it truly is, as seen below in Figure 13 and Figure 14. For example, the peach has a nominal error of 7.933 kcal which is almost negligible, with being roughly 0.003% of the daily required intake.

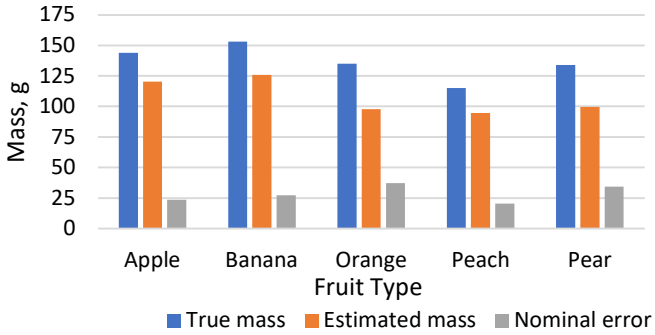


Figure 13 True mass compared to estimated mass

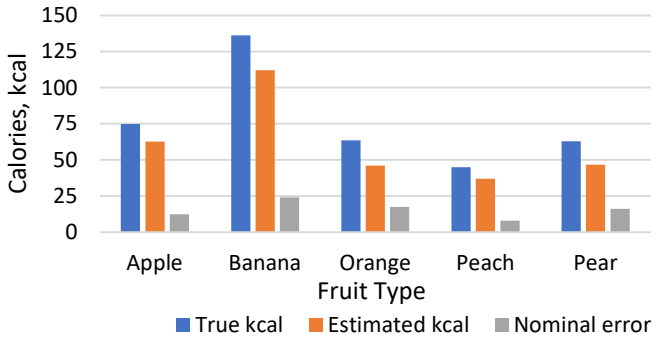


Figure 14 True calories compared to estimated calories

4.1.2. Mobile Testing

4.1.2.1. Image Acquisition

While the initial method was convenient, by not requiring any additional equipment, the subsequent images were not on the same plane, as seen in Figure 15, unlike those from lab testing, Figure 5. This resulted in erroneous pixels being detected or incorrectly located in space, resulting in a distorted point cloud. Often multiple attempts were required with some samples as the generated point clouds were corrupted. This was likely due to the calibration methods used and insufficient reference points for the software to conduct successful feature matching.

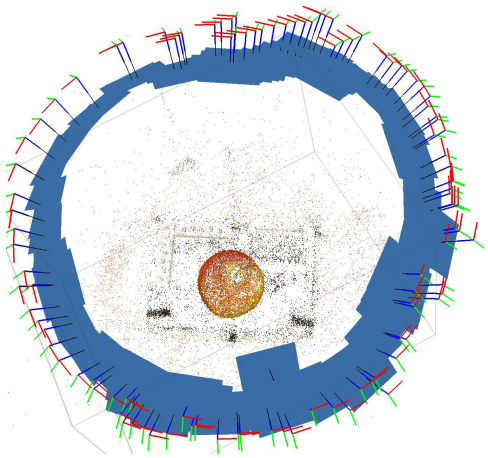


Figure 15 Mobile testing point cloud of an apple with relation to image planes.

Implementing a phone stand mitigated this issue as it allowed for a set object distance compared to above, while also removing the translation and rotational parameters imparted to the phone via the shaking of a phone when holding it while pictures are taken. In Figure 8, the phone stand is 134mm from a B5 notepad, this distance is dependent on the entire object being in frame while also ensuring sufficient clarity to identify the checkerboard for calibration. However, even though this approach is more reliable, it is still difficult to have the notepad rotate on the same point and not move laterally (as experienced during data collection) as this could cause problems during feature matching and impact the reliability of the camera's calibration.

4.1.2.2. Calibration Methods

A comparison between the calibration methods were undertaken, Figure 17, by following the appropriate procedures as outlined in 3.2.2.2. The lab testing results and uncalibrated reconstructions are used for comparison giving an upper and lower bounds of expected accuracy. The numerical data of each method can be found in the appendix.

While testing, probe calibration was found to be the most convenient method as any potential user would have a credit card or coins on their person as standard. Whereas the checkerboard and scale bar methods require a physical item that requires an additional item to be carried, increasing the burden of the potential user. This approach taken resulted in an inadequate, sparce, point cloud, seen below in Figure 16. Neither probe was capable of accurately nor reliably being measured due to the reconstructed geometry, voiding this calibration method and thus not included in this comparison.

Figure 16 Mobile probe object calibration point cloud with annotations.

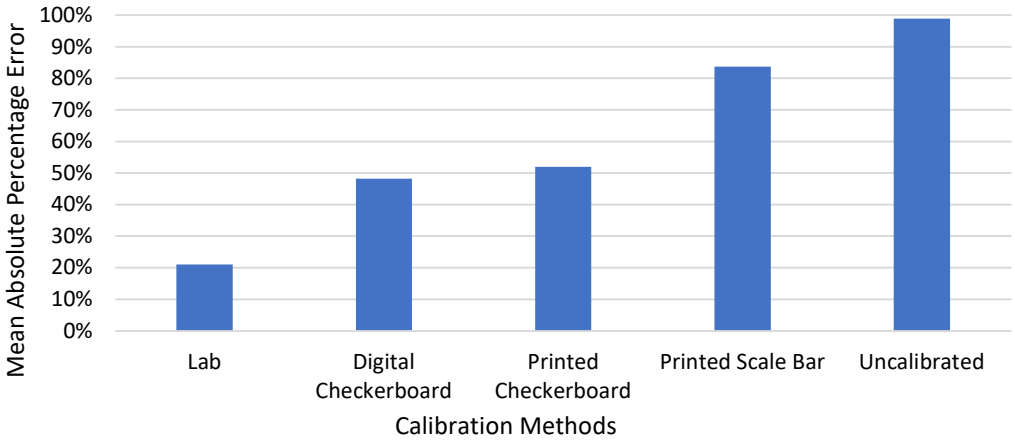


Figure 17 Percentage error comparing alternative calibration methods.

Using a digital checkerboard is also highly convenient as it is only required once, providing the distance between devices is constant which is difficult to distinguish and repeat. Although it is capable of calibrating the camera at multiple focal lengths compared to alternative methods where changing the focal length requires recalibration. Being the most accurate mobile calibration method by 3.7%, its greatest drawback is due to its advantage. If the calibration images and image acquisition are taken at greatly different distances the result cannot be relied upon to be continually accurate. Meanwhile using a printed checkerboard resulted in a more convenient and reliable method by comparison. Although there are negatives on using a printed checkerboard over a certified one, as discussed below, 4.1.3.

The scale bar resulted in the largest error rate from all calibration methods. Likely due to a few causes some of which were, only six coded targets were recognised out of the thirteen, seen in Figure 18, reducing the number of possible reference points that can be used to produce accurate scaling. Further, it is recommended to use a minimum of 3 scale bars [60], however when working on this scale there was insufficient space to do so, while ensuing enough points remain in frame over the acquisition processes. To include multiple scale bars the camera will need to be placed further away from target. While this would aid in increasing accuracy, having the target fruit further away would reduce the resulting reconstruction quality, thus this method was not pursued further.

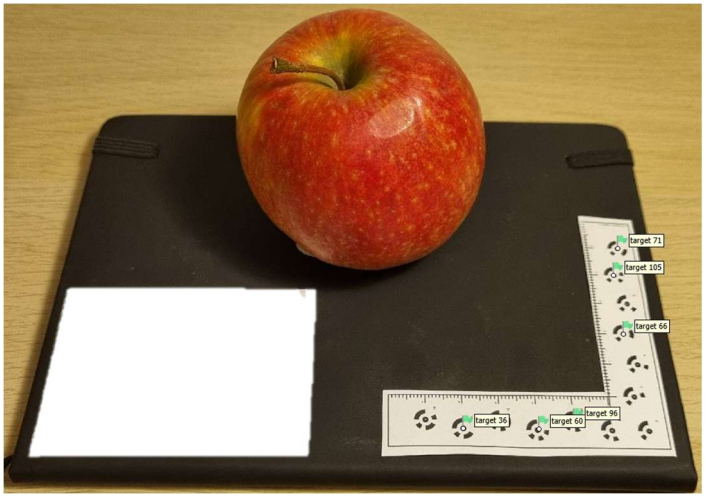


Figure 18 Identified coded targets of a printed scale bar during image acquisition.

4.1.3. Volume & Calorie Calculation

A similar approach to calorie calculation seen in 4.1.1 using Equation 3 was taken. The deviation between methods is due to each fruit variety requiring a non-constant conversion factor, to change a unitless volume into meters cubed. Table 3 contains the results of using the second image acquisition method with the printed checkerboard. As it uses the same parameter calibration code as 3.2.1, the generated volume from this is to be assumed in millimetres cubed. However, it isn't the case as observed when comparing the pixel volume magnitudes to each other and to Table 1. A pixel volume with the magnitude of 10,000 is accepted to be in millimetres due to the accuracy and reliability of the lab results and methods used. Hence, each fruit is scaled in that respect as seen below, asides from the pear in which a greater scale is required because of its organic shape.

The need for variable scaling is likely a result of the image acquisition and calibration stage of the process. Using the printed checkerboard over a manufactured one carries significant risk. It was not lying flat during the acquisition, Figure 9, which imparted improper translation and rotation values into the result. The vantage of the phone impacts the clarity of the checkerboard, with further squares being more pixilated and thus determining the vertex to a high enough degree increases complexity. The vantage point is a difficult variable to manage, as a shallow angle allows for larger objects to fit in the frame but reduces the calibration image quality. While a larger angle aids in calibration, it reduces the number of detected features, with a higher view increasing shadows and probability of obstructed geometry.

Fruit Type	Pixel volume	Scale value	True, g	Estimated, g	True, $kcal$	Estimated, $kcal$	Error, %
Apple	5,199.415	1e+8	144	49.40	77.48	25.69	66.85
Banana	21.783	1e+5	153	239.61	130.83	213.26	57.33
Orange	820.221	1e+7	135	73.82	81.31	34.70	57.33
Peach	11.697	1e+5	115	116.74	37.83	45.53	20.35
Pear	75.5580	1e+6	134	78.28	76.61	36.79	51.98
						Mean absolute percentage error	51.90

Table 3 Estimated calories and percentage error of fruit via printed checkerboard and phone stand.

The mobile close-range photogrammetry pipeline is compared against current apps on the market, Figure 19, with data in the appendix. MyFitnessPal [8] and NutraCheck [10] require the user to weigh food and manually select its variety, while SnapCalorie [12] uses AI to identify both the size and type of food via a single image using their custom algorithm [61]. In these tests, all apps were used as either a free trial or on the free version of the app as a baseline comparison, as occasionally paid apps come with increased features that can lead to higher accuracy.

Lab testing having a lower MAPE of 21% than SnapCalorie's 30% supports the use and further investigation of using photogrammetry for nutritional tracking. The difference between Nutracheck and MyFitnessPal are likely due to the calculations and the data which they use, however being a closed app it is not possible to determine why there's such a disparity between them. While MyFitnessPal is the most accurate nutritional tracking app, it is cumbersome, leaving an opening in the market for the proposed pipeline. Further, lab testing was yet more accurate than NutraCheck, which disputes the notion that using a scale will result in more accurate results.

The mobile data is from the printed checkerboard calibration method, in which the reasons for large inaccuracies are addressed above. The accuracy of the peach is on par with the other options, aside from SnapCalorie, hence demonstrating mobile photogrammetry is viable for this application but further work is required; addressed in section 6.

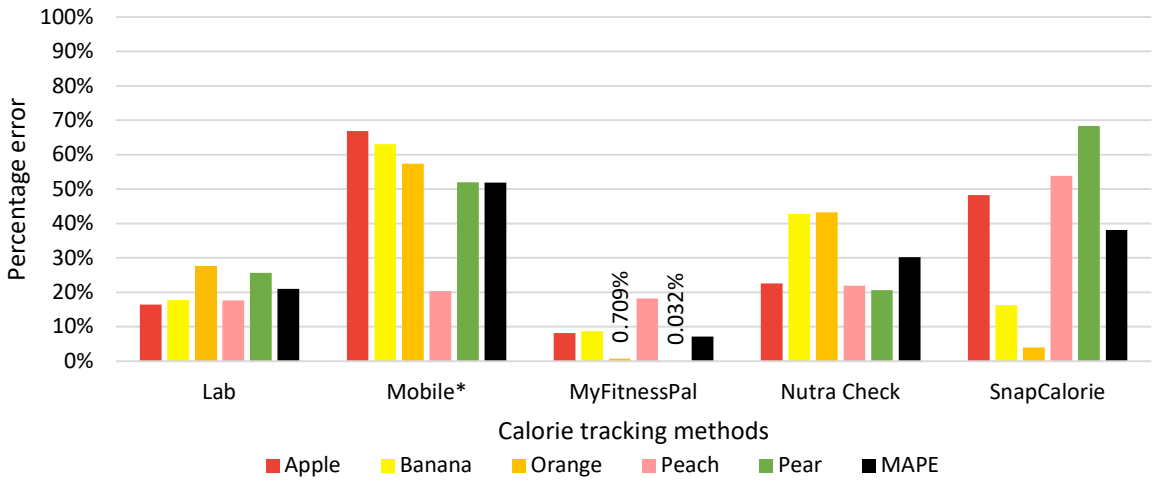


Figure 19 Comparison in percentage error between pipeline tests and commercial alternatives.

Mobile testing was conducted with different test fruit to the others because of the timeline. However, the size and ripeness remained near identical to reduce any inaccuracies. The numerical results can be found in the appendix.

4.2. Food Recognition

4.2.1. Image Recognition

Each of the chosen image recognition models, as outlined in 3.3.1, were tested with the three images seen below in Figure 20; an orange, an apple and several mangoes. The models were not tested with an image from the dataset used for training to avoid overfitting, and in the effort of being representative of realistic use, as the actual fruit will not be identical to those in the dataset.


[62]

[63]

[64]

Figure 20 Image recognition test images

The models are tested against two metrics: accuracy and speed, as seen below in Figure 21. Vision API had the greater average accuracy of 88% with Vertex AI being slightly less at 74%. This is contradictory to the norm, as commonly, when performing transfer training, accuracy is expected to increase but this may be

because of the dataset used. As seen below, the custom model data was inconclusive. The mean percentage error was 77% however, it falsely predicted the class of images. Where it predicted an apple was a tomato, mangos an orange, and an orange was a banana. False classification would be a result of training, and the dataset used. The dataset consisted of images of a very low quality, 320×258 pixels, so the model was trained with a small image size. Thus, the test images were downscaled to a similar size, as with computer vision the image sizes must be constant from training to deployment, otherwise the model would not read the image. Scaling the test images would cause distortion and pixelation, the cause of false classification.

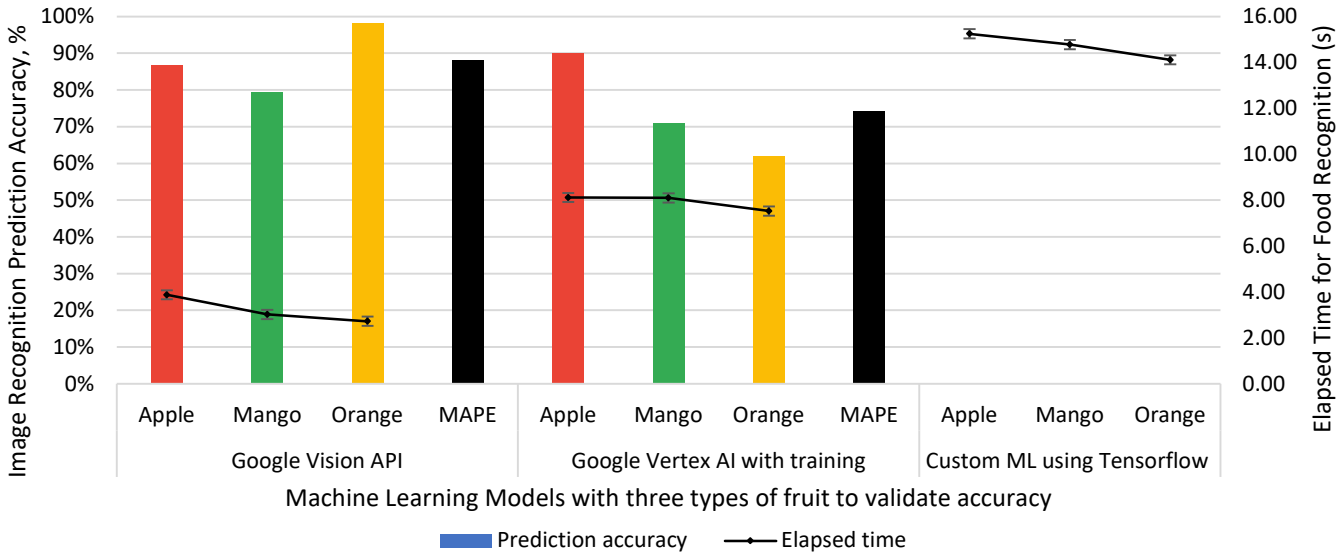


Figure 21 different image recognition models were provided with 3 different images to evaluate prediction accuracy and elapsed time.

4.2.2. Object Detection

Pretrained YOLO11n uses the COCO, common objects in context, dataset [65], and while it contains 80 pre-trained classes, only three of these were fruit; apple, banana and orange. Although it provided a high classification accuracy, averaging 86% for those identified correctly, it incorrectly identified a peach and pear as an orange, Figure 22. This is clearly due to an insufficient training dataset, thus the dataset required expanding. Creation of a manual dataset was attempted but due to time constraints, and issues encountered with image scraping APIs, alternate large object detection datasets were sought out.

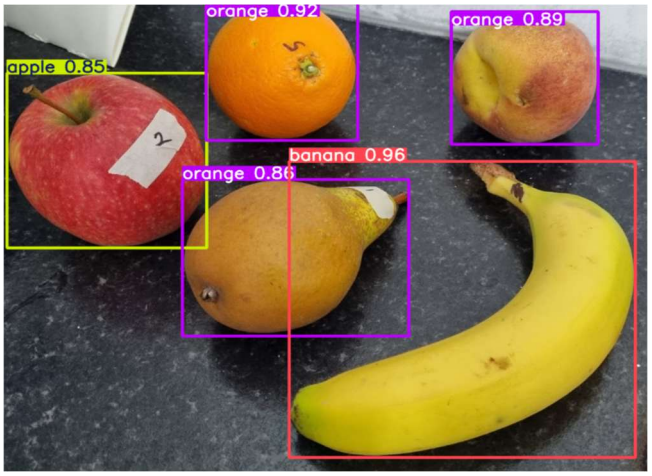


Figure 22 Graphical representation, OD with COCO dataset of the fruits bounding boxes and prediction accuracy.

Open Images v7, funded by Google, contains over nine million images with segmentation and labelling [66]. However, as seen in Figure 23, the accuracy was lower than with COCO but also incorrectly identifies an orange as a cantaloupe and a peach as an apple. Similarly, this is due to a poor-quality dataset for fruit recognition, with only 497 images of fruit at sixteen classes, with a heavy skew towards apple, thus more similar fruit will be falsely recognised as an apple than its actual type. There is insufficient training data for the pear to be registered as any fruit type, hence, there is no bounding box, even though the dataset does contain such information and that it should correctly identify all fruit in the image.

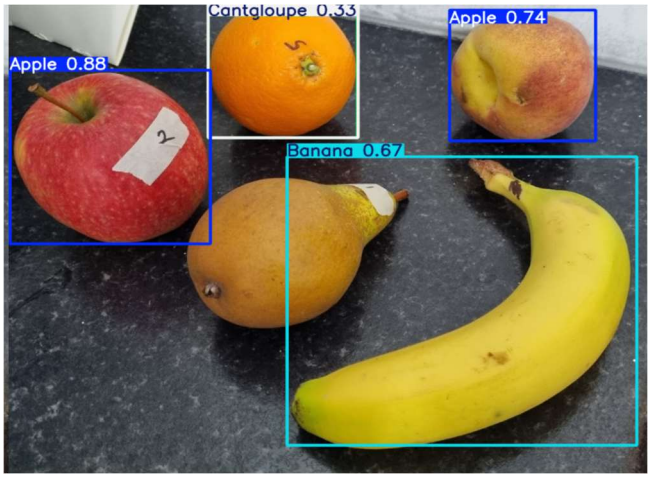


Figure 23 Graphical representation, OD with Open Images v7 dataset of the fruits bounding boxes and prediction accuracy.

A further dataset investigated was LVIS, large vocabulary instance segmentation, created by Facebook AI Research for the purpose of object detection [67]. It consists of one hundred thousand images for training alone, containing 180 images of fruit, of which there are 41 different varieties. Thus, its weightings are heavily skewed to those classes with the most training images, in LVIS's case apple and banana have 12 and 32 images respectively. The fruit correctly identified had a higher average prediction accuracy of 94% compared to the 86% of COCO, making this dataset more reliable, Figure 24. However, like previous options, LVIS is unable to accurately detect a pear or a peach, likely due to having an average of only 4 images per class. As a peach has visual similarities between that and an apple, in colour and shape, resolving this can be achieved by improving the dataset in number of images and quality.

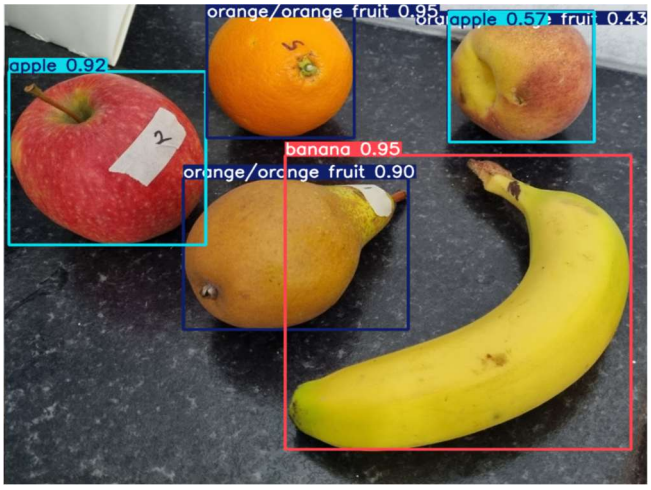


Figure 24 Graphical representation OD with LVIS dataset of the fruits bounding boxes and prediction accuracy.

Falsely identifying multiple objects as an orange can signify multiple issues. Primarily it is the result of a poor image dataset for the intended application, but it may also be a sign of poor training. Training quality can be

impacted by the computer hardware it is trained on. Performing training on a GPU allows for faster computation over a CPU, going from multiple days to a couple hours, but is limited by the video random access memory, VRAM. Having insufficient VRAM directly impacts image resolution, batch size, and can limit the complexity of model's parameters to avoid crashing [50].

After training YOLO11n on the LVIS dataset a normalised matrix is generated, it is a graphical method of evaluating the performance of predictions against each other. It should have a solid diagonal line, where the predicted value and true value always meet. Some variance is accepted however; in Figure 25 it is significantly non-linear with the background being predicted more than the actual fruit. As outlined above this is likely due to an insufficient dataset and possible hardware limitations.

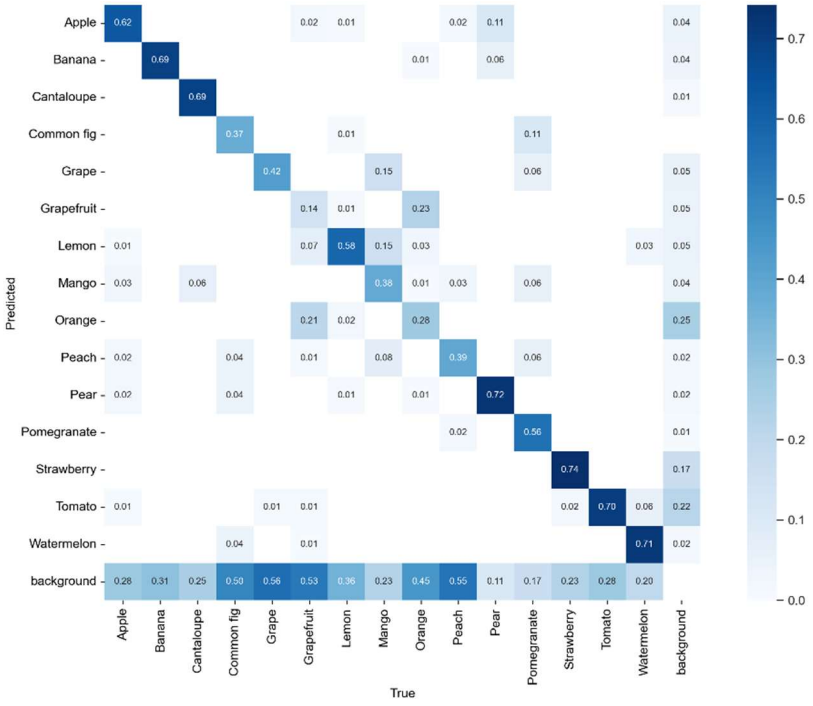


Figure 25 Normalised confusion matrix of YOLO11n with a subsection of the LVIS dataset.

Loss measures the difference between a model's prediction and the true result [50]. Training was commuted after converging for five iterations, with the aim of reducing the potential of overfitting. Figure 26 shows training loss, how well the model is performing and fitting the training data. The loss shouldn't be too low as it can signify overfitting of the training data. Where the model isn't learning it is just memorising the answers to the training data and generating a detrimental feedback loop. Validation loss represents how accurate the model is on a separate set of data, evaluating how the model works overall. In Figure 27, validation loss starts to converge however, as class loss keeps falling, this is a potential sign of overfitting, providing evidence behind the incorrect fruit detection seen in Figure 24.

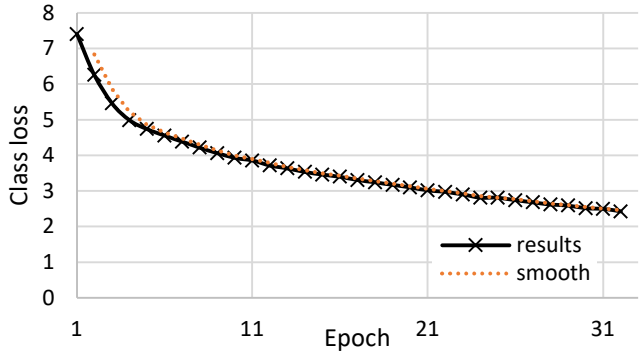


Figure 26 YOLO11n class loss during LVIS training

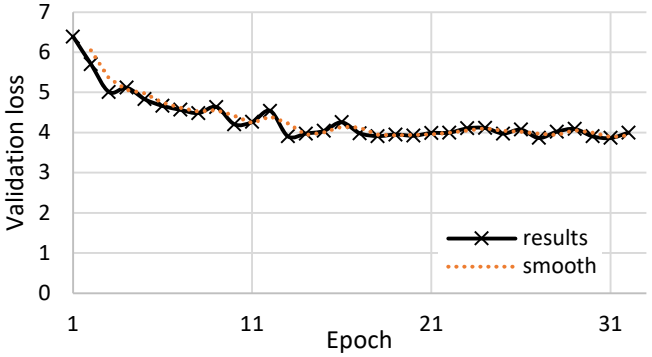


Figure 27 YOLO11n validation loss during LVIS training

5. Conclusions

Photogrammetry is a measurement technique that has its uses in many fields but has not been fully realised with food. This project has developed a pipeline to calculate calories of food via close-range photogrammetry and food recognition. Results from the laboratory testing determined that the pipeline is viable, with a 21% MAPE, to use close-range photogrammetry in calculating foods calories.

Mobile testing was conducted to explore how the pipeline can be implemented into an application. Different calibration methods were compared to identify the most accurate and efficient way to calibrate a phone, where a printed checkerboard was the leading method with 51.9% MAPE. Existing applications may be more accurate but, considering the practicality of weighing food for nutritional tracking, this pipeline is viable as an application. This report has established significant research in alternative mobile calibration methods and indicated, with further research, the pipeline will become more accurate than alternate nutritional tracking methods.

Object detection is the most appropriate method for food recognition as it provides bounding boxes, allowing for the implementation of calculating multiple foods at once. As seen in this report, the dataset used in training a deep learning model is vital, as using a bad dataset is equally influential in providing poor results, as with bad training.

6. Further Improvements

The pipeline in its current state still requires significant human interaction to go from taking images to obtaining calories of the fruit. In the effort of reducing human burden, Agisoft Metashape is capable of script automation, thus zero interactions are necessary to generate a dense point cloud. However, as of writing this project there is no such automation for cleaning up a point cloud within CloudCompare.

As identified in 4.1.2.2, a physical checkerboard can generate accurate data but variability in the mobile image acquisition set up led to inaccuracies. Further research into accurate and repeatable mobile calibration methods is required.

Additionally, the ML dataset used must be improved upon. A larger dataset consisting of a reasonable number of images per class, along with a wider range, will increase accuracy of the model and hence reduce the errors in calories. This can be achieved by creating a custom dataset or via consolidating multiple existing datasets.

7. References

[1] B. Bogin, "From caveman cuisine to fast food: the evolution of human nutrition," *Growth Hormone and IGF research,* vol. 8, pp. 79-88, 1998.

[2] R. I. M. Dunbar, "Breaking Bread: the Functions of Social Eating," *Adaptive Human Behavior and Physiology,* vol. 3, no. 3, pp. 198-211, 2017/09/01 2017, doi: 10.1007/s40750-017-0061-4.

[3] "Understanding calories." NHS. https://www.nhs.uk/live-well/healthy-weight/managing-your-weight/understanding-calories/ (accessed 24/10, 2024).

[4] C. D. Economos, S. S. Bortz, and M. E. Nelson, "Nutritional Practices of Elite Athletes," *Sports Medicine,* vol. 16, no. 6, pp. 381-399, 1993/12/01 1993, doi: 10.2165/00007256-199316060-00004.

[5] M. Muscaritoli *et al.*, "ESPEN practical guideline: Clinical Nutrition in cancer," *Clinical Nutrition,* vol. 40, no. 5, pp. 2898-2913, 2021/05/01/ 2021, doi: https://doi.org/10.1016/j.clnu.2021.02.005.

[6] Y. Hasegawa *et al.*, "Protein intake after the initiation of chemotherapy is an independent prognostic factor for overall survival in patients with unresectable pancreatic cancer: A prospective cohort study," *Clinical Nutrition,* vol. 40, no. 7, pp. 4792-4798, 2021/07/01/ 2021, doi: https://doi.org/10.1016/j.clnu.2021.06.011.

[7] C. A. Levinson, L. Fewell, and L. C. Brosof, "My Fitness Pal calorie tracker usage in the eating disorders," *Eating Behaviors,* vol. 27, pp. 14-16, 2017/12/01/ 2017, doi: https://doi.org/10.1016/j.eatbeh.2017.08.003.

[8] *MyFitnessPal: Food & Fitness*. (2010). MyFitnessPal, Inc, Google Play. [Online]. Available: https://play.google.com/store/apps/details?id=com.myfitnesspal.android&hl=en

[9] *Foodvisor - Nutrition & Diet*. (2024). Foodvisor, Google Play. [Online]. Available: https://play.google.com/store/apps/details?id=io.foodvisor.foodvisor&gl=GB

[10] *Calorie Counter +*. (2024). Nutracheck, Google Play. [Online]. Available: https://play.google.com/store/apps/details?id=com.nutratech.app.android

[11] R. E. Brown *et al.*, "Calorie Estimation in Adults Differing in Body Weight Class and Weight Loss Status," *Medicine & Science in Sports & Exercise,* vol. 48, no. 3, 2016.

[12] *SnapCalorie AI Calorie Counter*. (2024). Perception Labs Inc, Google Play. [Online]. Available: https://play.google.com/store/apps/details?id=com.snapcalorie.alpha002&gl=GB

[13] T. Fanyu Kong and Jindong, "DietCam: Automatic dietary assessment with mobile camera phones," *Pervasive and Mobile Computing,* vol. 8, no. 1, pp. 147-163, 2012, doi: https://doi.org/10.1016/j.pmcj.2011.07.003.

[14] Y. Ando, E. Takumi, C. Jaehyeong, and Y. Keiji, "DepthCalorieCam: A Mobile Application for Volume-Based FoodCalorie Estimation using Depth Cameras," presented at the Proceedings of the 5th International Workshop on Multimedia Assisted Dietary Management, 2019. [Online]. Available: https://doi.org/10.1145/3347448.3357172.

[15] M. H. Rahman *et al.*, "Food Volume Estimation in a Mobile Phone Based Dietary Assessment System," 2012.

[16] M. Puri, Z. Zhu, Y. Qian, D. Ajay, and S. Harpreet, "Recognition and volume estimation of food intake using a mobile device," 2009.

[17] J. Dehais, A. Marios, S. Sergey, and M. Stavroula, "Two-View 3D Reconstruction for Food Volume Estimation," *IEEE Transactions on Multimedia,* vol. 19, no. 5, pp. 1090-1099, 2017, doi: 10.1109/TMM.2016.2642792.

[18] Y. Yue, W. Jau, and M. Sun, "Measurement of food volume based on single 2-D image without conventional camera calibration," presented at the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Annual International Conference, 2012.

[19] T. Luhmann, "Close range photogrammetry for industrial applications," *ISPRS Journal of Photogrammetry and Remote Sensing,* vol. 65, no. 6, pp. 558-569, 2010/11/01/ 2010, doi: https://doi.org/10.1016/j.isprsjprs.2010.06.003.

[20] T. Luhmann, S. Robson, S. Kyle, and J. Boehm, *Close-Range Photogrammetry and 3D Imaging*, 4th ed. Berlin/Boston: Walter de Gruyter GmbH & Co KG, 2023, p. 852.

[21] T. Luhmann, C. Fraser, and H.-G. Maas, "Sensor modelling and camera calibration for close-range photogrammetry," *ISPRS Journal of Photogrammetry and Remote Sensing,* vol. 115, pp. 37-46, 2016/05/01/ 2016, doi: https://doi.org/10.1016/j.isprsjprs.2015.10.006.

[22] P. Patonis, "A Comparative Study on the Use of Smartphone Cameras in Photogrammetry Applications," *Sensors*, vol. 24, no. 22, doi: 10.3390/s24227311.

[23] M. Shortis, C. Bellman, S. Robson, G. J. Johnston, and G. W. Johnson, "Stability of zoom and fixed lenses used with digital SLR cameras," *Int. Arch. Photogramm. Remote Sens.,* vol. 36, 11/30 2005.

[24] T. N. Ataiwe, I. Hatem, and H. M. J. Al Sharaa, "Digital Model in Close-Range Photogrammetry Using a Smartphone Camera," *E3S Web Conf.,* 10.1051/e3sconf/202131804005 vol. 318, // 2021.

[25] *Agisoft Metashape*. (2024). Agisoft. [Online]. Available: https://www.agisoft.com/downloads/installer/
[26] A. H. Qureshi, W. S. Alaloul, A. Murtiyoso, S. Saad, and B. Manzoor, "Comparison of Photogrammetry Tools Considering Rebar Progress Recognition," *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.,* vol. Volume XLIII-B2-2022, pp. 141–146, 2022, doi: 10.5194/isprs-archives-XLIII-B2-2022-141-2022.
[27] X. q. Li, Z. a. Chen, L. t. Zhang, and D. Jia, "Construction and Accuracy Test of a 3D Model of Non-Metric Camera Images Using Agisoft PhotoScan," *Procedia Environmental Sciences,* vol. 36, pp. 184-190, 2016/01/01/ 2016, doi: https://doi.org/10.1016/j.proenv.2016.09.031.
[28] L. Jurjević and M. Gašparović, "3D Data Acquisition Based on OpenCV for Close-Range Photogrammetry Applications," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences,* vol. XLII-1/W1, pp. 377-382, 2017, doi: 10.5194/isprs-archives-XLII-1-W1-377-2017.
[29] E. Alpaydin, *Machine Learning: The New AI* (The MIT Press Essential Knowledge series). MIT Press, 2016.
[30] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science,* vol. 313, no. 5786, pp. 504-507, 2006/07/28 2006, doi: 10.1126/science.1127647.
[31] W. Meiyin and C. Li, "Image recognition based on deep learning," in *2015 Chinese Automation Congress (CAC)*, 27-29 Nov. 2015 2015, pp. 542-546, doi: 10.1109/CAC.2015.7382560.
[32] G. Ciocca, P. Napoletano, and R. Schettini, "Food Recognition: A New Dataset, Experiments, and Results," *IEEE Journal of Biomedical and Health Informatics,* vol. 21, no. 3, pp. 588-598, 2017, doi: 10.1109/JBHI.2016.2636441.
[33] A. Bailly *et al.*, "Effects of dataset size and interactions on the prediction performance of logistic regression and deep learning models," *Computer Methods and Programs in Biomedicine,* vol. 213, p. 106504, 2022/01/01/ 2022, doi: https://doi.org/10.1016/j.cmpb.2021.106504.
[34] K. Moumane, I. E. Asri, T. Cheniguer, and S. Elbiki, "Food Recognition and Nutrition Estimation using MobileNetV2 CNN architecture and Transfer Learning," in *2023 14th International Conference on Intelligent Systems: Theories and Applications (SITA)*, 22-23 Nov. 2023 2023, pp. 1-7, doi: 10.1109/SITA60746.2023.10373725.
[35] *digiCamControl*. (2025). [Online]. Available: https://digicamcontrol.com/
[36] U. o. Nottingham. "Manufacturing Metrology Team." https://www.nottingham.ac.uk/research/groups/advanced-manufacturing-technology-research-group/research/manufacturing-metrology-team/index.aspx (accessed May, 2025).
[37] *Camera Calibrator MATLAB*. (2013). The Mathworks Inc.
[38] *CloudCompare*. (2024).
[39] "Samsung Galaxy S21+ 5G SD888 - Specifications." DeviceSpecifications. https://www.devicespecifications.com/en/model/dfb45528 (accessed May, 2025).
[40] "Checkerboard Calibration Targets On Ceramic." Shenzhen Visions Plus Co. Ltd. https://www.testargets.com/checkerboard-target/high-precision-machine-vision-checkerboard/checkerboard-calibration-targets-on-ceramic.html (accessed May, 2025).
[41] "Pattern Generator." calib.io. https://calib.io/pages/camera-calibration-pattern-generator (accessed May, 2025).
[42] "Lens calibration (using chessboard pattern) in Metashape." https://agisoft.freshdesk.com/support/solutions/articles/31000160059-lens-calibration-using-chessboard-pattern-in-metashape (accessed May, 2025).
[43] "Precision Measurement with Target Marks." PMS AG Switzerland. https://en.elcovision.com/elcovison-10-zielmarken.html (accessed May, 2025).
[44] J. Brownlee, *Deep learning for computer vision: image classification, object detection, and face recognition in python*. Machine Learning Mastery, 2019.
[45] *Classify 15 fruits with tensorflow accuracy 99.6%*. (2021). kaggle.com. [Online]. Available: https://www.kaggle.com/code/databeru/classify-15-fruits-with-tensorflow-acc-99-6
[46] *BEng Project-Improving Nutritional Tracking Code*. (2024). Github. [Online]. Available: https://github.com/NoahSantaub/BEng_Project-Improving_Nutritional_Tracking
[47] R. Joseph, D. Santosh, G. Ross, and F. Ali, "You Only Look Once: Unified, Real-Time Object Detection," 2016.
[48] R. Shaoqing, H. Kaiming, G. Ross, and S. Jian, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 39, no. 6, pp. 1137-1149, 2017, doi: 10.1109/TPAMI.2016.2577031.
[49] K. Sawant *et al.*, "Fruit Identification using YOLO v8 and Faster R-CNN –A Commparative Study," 2024.

[50] *Ultralytics YOLO11*. (2024). [Online]. Available: https://github.com/ultralytics/ultralytics[Online]. Available: https://docs.ultralytics.com/models/yolo11/
[51] N. Jegham, C. Y. Koh, M. Abdelatti, and A. Hendawi, "Evaluating the Evolution of YOLO (You Only Look Once) Models: A Comprehensive Benchmark Study of YOLO11 and Its Predecessors," *arXiv preprint arXiv:2411.00201,* 2024.
[52] *FoodData Central, https://fdc.nal.usda.gov/*,
[53] *FooDB, Version 1.0, www.foodb.ca*,
[54] *Nutritionix Database, Common Foods*, Syndigo Company,
[55] I. Ozturk, S. Bastaban, S. Ercisli, and F. Kalkan, "Physical and chemical properties of three late ripening apple cultivars," *International Agrophysics,* vol. 24, no. 4, pp. 357-361, 2010.
[56] K. R. Pawar, V. T. Atkari, and R. F. Sutar, "Engineering properties of raw banana (Musa paradisica L.) fruit," 2012.
[57] A. Topuz, M. Topakci, M. Canakci, I. Akinci, and F. Ozdemir, "Physical and nutritional properties of four orange varieties," *Journal of Food Engineering,* vol. 66, no. 4, pp. 519-523, 2005/02/01/ 2005, doi: https://doi.org/10.1016/j.jfoodeng.2004.04.024.
[58] A. Pérez-López, S. H. Chávez-Franco, C. A. Villaseñor-Perea, T. Espinosa-Solares, L. H. Hernández-Gómez, and C. Lobato-Calleros, "Respiration rate and mechanical properties of peach fruit during storage at three maturity stages," *Journal of Food Engineering,* vol. 142, pp. 111-117, 2014/12/01/ 2014, doi: https://doi.org/10.1016/j.jfoodeng.2014.06.007.
[59] J. Wang, "Mechanical Properties of Pear as a Function of Location and Orientation," *International Journal of Food Properties,* vol. 7, no. 2, pp. 155-164, 2004/12/31 2004, doi: 10.1081/JFP-120025392.
[60] C. H. Imaging, "Guidelines for Calibrated Scale Bar Placement and Processing," vol. Version 2.0, ed. https://www.agisoft.com/pdf/tips_and_tricks/CHI_Calibrated_Scale_Bar_Placement_and_Processing.pdf, 2015.
[61] Q. Thames *et al.*, "Nutrition5k: Towards automatic nutritional understanding of generic food," 2021, pp. 8903-8911.
[62] orenfoods. "small orange image." https://orenfoods.co.uk/cy/shop/small-orange (accessed May, 2025).
[63] G. F. Guide. "Apple - Bicoloured." https://goodfruitguide.co.uk/product/zari/ (accessed May, 2025).
[64] M. Kozlenko, "Mangoes on sale in Australia," ed. wikimedia, 2016.
[65] T.-Y. Lin *et al.*, "Microsoft COCO: Common Objects in Context," in *Computer Vision – ECCV 2014*, Cham, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., 2014// 2014: Springer International Publishing, pp. 740-755.
[66] A. Kuznetsova *et al.*, "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale," *International journal of computer vision,* vol. 128, no. 7, pp. 1956-1981, 2020.
[67] G. Agrim, D. Piotr, and G. Ross, "LVIS: A Dataset for Large Vocabulary Instance Segmentation," 2019.

8. Appendix

Close Range Photogrammetry Point Clouds

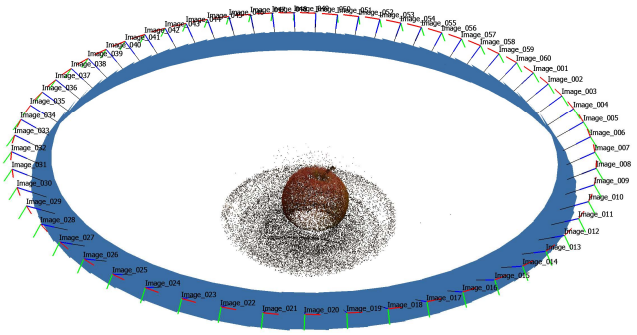
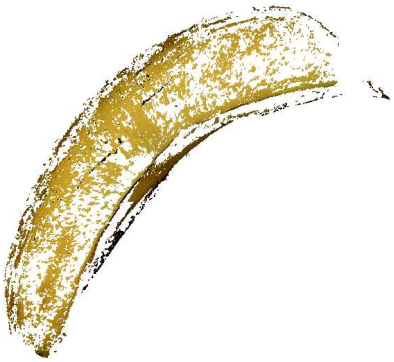
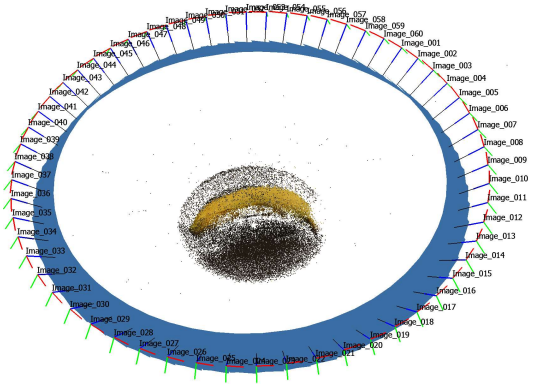
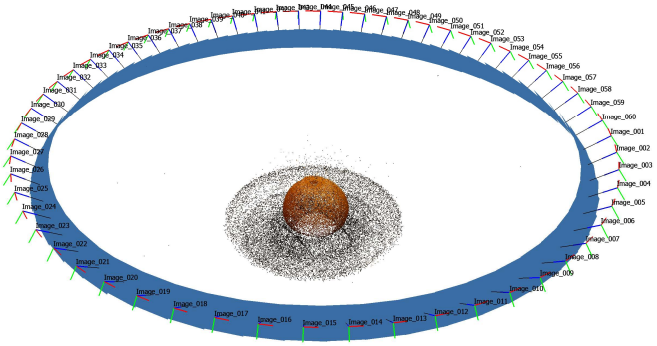
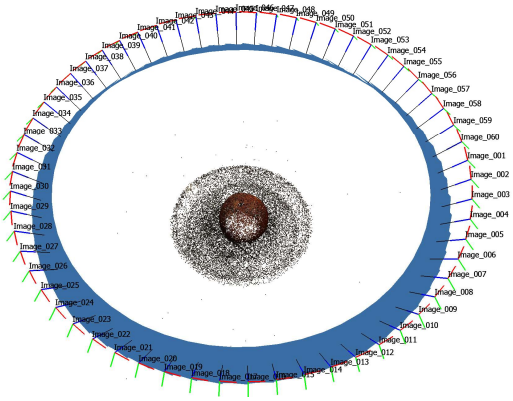
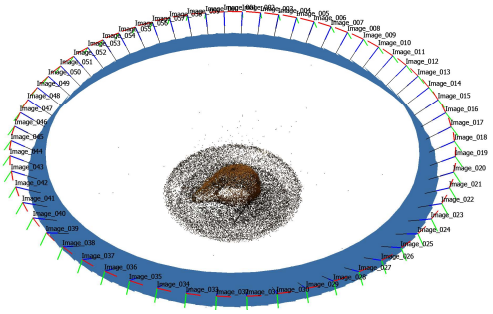
Type	Cleaned point cloud	Agisoft Metashape point cloud reconstruction
Apple		
Banana		
Orange		
Peach		
Pear		

Table 4 Laboratory Testing Point Clouds

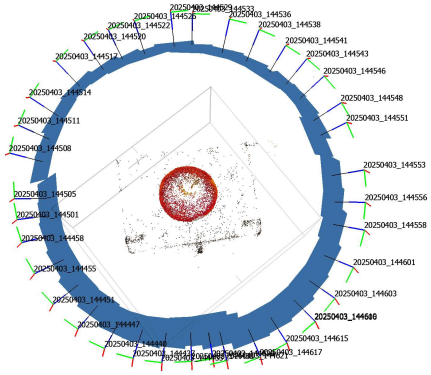
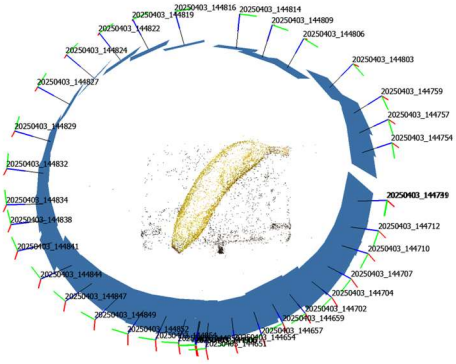
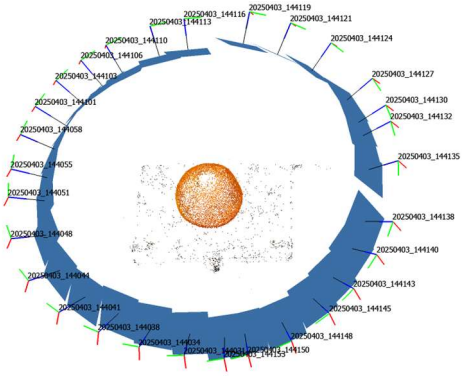
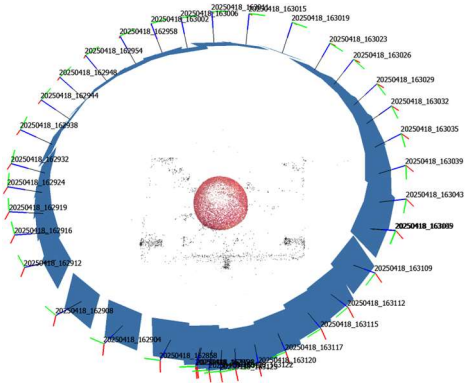
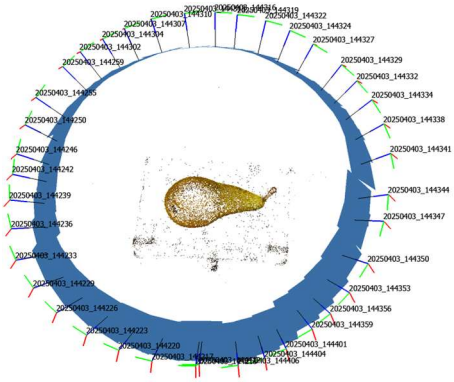
Type	Cleaned point cloud	Agisoft Metashape point cloud reconstruction
Apple		
Banana		
Orange		
Peach		
Pear		

Table 5 Mobile Testing Point Clouds, Printed Checkerboard

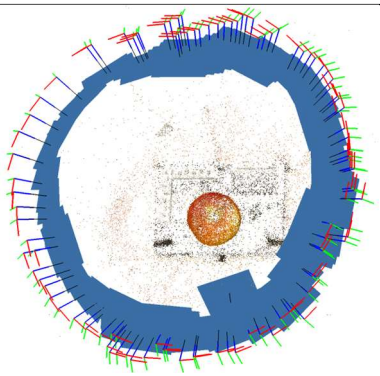
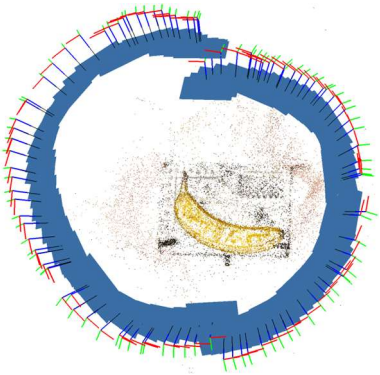
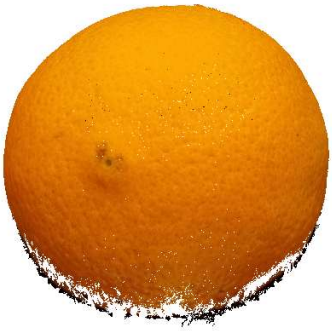
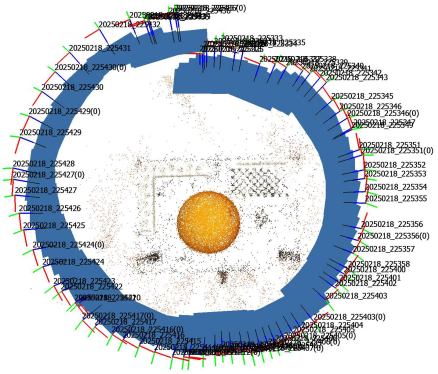
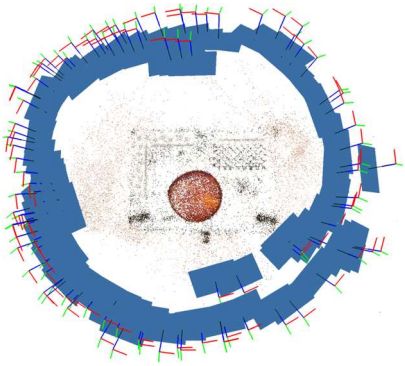
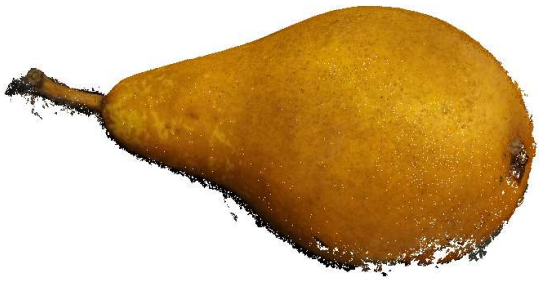
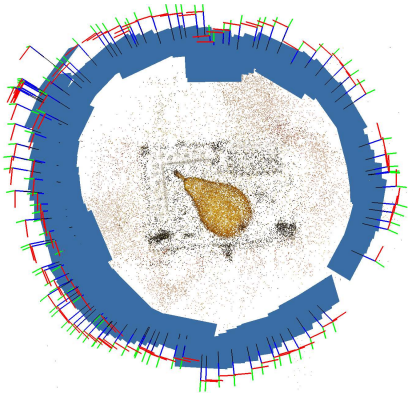
Type	Cleaned point cloud	Agisoft Metashape point cloud reconstruction
Apple		
Banana		
Orange		
Peach		
Pear		

Table 6 Mobile Testing Point Clouds, Digital Checkerboard

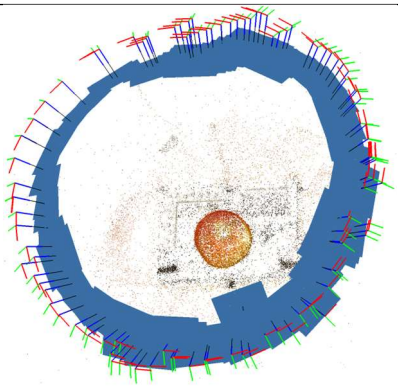
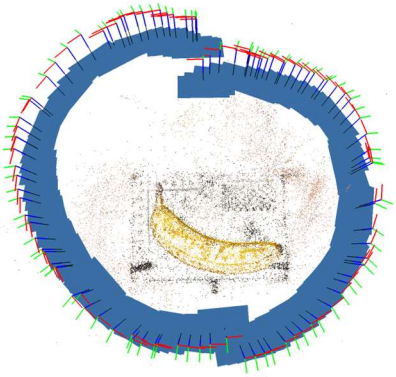
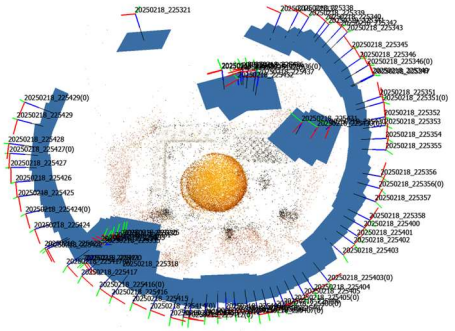
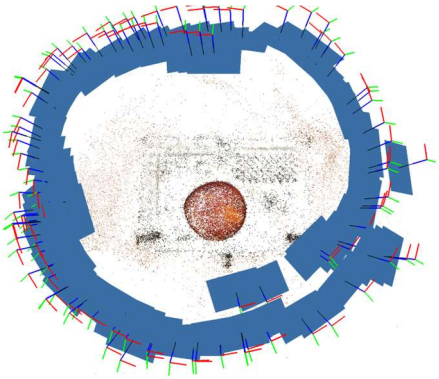
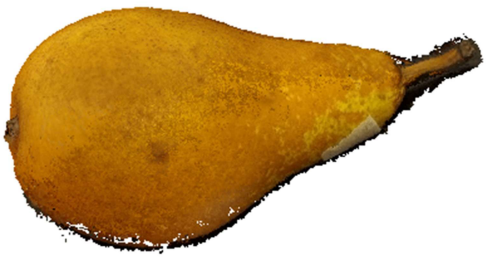
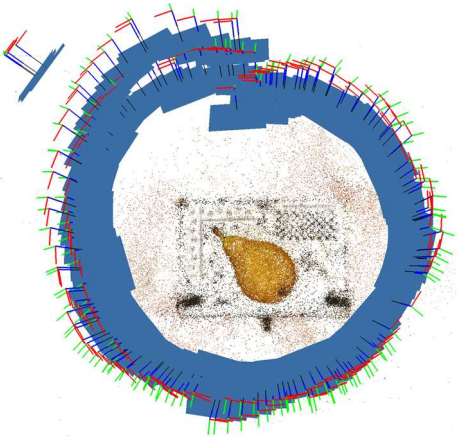
Type	Cleaned point cloud	Agisoft Metashape point cloud reconstruction
Apple		
Banana		
Orange		
Peach		
Pear		

Table 7 Mobile Testing Point Clouds, Printed Scale Bar

Close Range Photogrammetry Data

Fruit Type	Pixel volume	Scale value	True, g ±1	Estimated, g	Nominal Error	True, $kcal$	Estimated, $kcal$	Nominal Error	Error, %
Apple	126,694.701	1e+9	144	120.360	23.640	74.880	62.587	12.293	16.417
Banana	192,812.998	1e+9	153	125.819	27.181	126.170	111.979	24.191	17.765
Orange	108,673.713	1e+9	135	97.806	37.194	63.450	45.969	17.481	27.551
Peach	94,848.731	1e+9	115	94.659	20.341	44.850	36.917	7.933	17.688
Pear	96,160.000	1e+9	134	99.622	34.378	62.980	46.822	16.158	25.656
								Mean absolute percentage error	21.015

Table 8 Lab testing mass and volume estimation

Fruit Type	Pixel volume	Scale value	True, g ±1	Estimated, g	Nominal Error	True, $kcal$	Estimated, $kcal$	Nominal Error	Error, %
Apple	5,199.415	1e+8	149	49.394	99.605	77.480	25.658	51.795	66.849
Banana	21.783	1e+5	147	239.613	92.613	130.830	213.256	82.426	63.002
Orange	820.221	1e+7	173	73.820	99-180	81.310	34.695	46.615	57.329
Peach	11.697	1e+5	97	116.736	46.264	37.830	45.537	7.697	20.346
Pear	75.558	1e+6	163	78.278	84.722	76.610	36.791	39.819	51.977
								Mean absolute percentage error	51.901

Table 9 Mobile testing with printed checkerboard, mass and volume estimation

Fruit Type	Pixel volume	Scale value	True, g ±1	Estimated, g	Nominal Error	True, $kcal$	Estimated, $kcal$	Nominal Error	Error, %
Apple	2,620.423	1e+8	144	24.894	119.106	74.88	12.945	61.935	82.712
Banana	570.395	1e+7	153	62.743	90.257	126.17	55.842	80.328	58.991
Orange	4,557.627	1e+8	135	41.019	93.981	63.45	19.279	44.171	69.616
Peach	9.383	1e+5	115	93.642	21.358	44.85	36.521	8.329	18.572
Pear	115.199	1e+6	134	119.346	14.654	62.98	56.093	6.887	10.935
								Mean absolute percentage error	48.165

Table 10 Mobile Testing with digital checkerboard, mass and volume estimation

Fruit Type	Pixel volume	Scale value	True, g ±1	Estimated, g	Nominal Error	True, $kcal$	Estimated, $kcal$	Nominal Error	Error, %
Apple	0.167	1000	144	158.650	14.650	77.480	82.498	7.618	10.174
Banana	0.330	1000	153	363.000	210.000	130.830	323.070	186.900	137.255
Orange	0.349	1000	135	314.100	179.100	81.310	147.627	84.177	132.667
Peach	0.173	1000	115	172.654	50.134	37.830	67.335	22.485	50.134
Pear	0.240	1000	134	251.758	87.872	76.610	118.322	55.342	87.872
								Mean absolute percentage error	83.620

Table 11 Mobile testing with printed scale bar, mass and volume estimation

Fruit Type	Pixel volume	Scale value	True, g ±1	Estimated, g	Nominal Error	True, $kcal$	Estimated, $kcal$	Nominal Error	Error, %
Apple	2,835.927	1e+9	144	2.694	141.306	77.480	1.401	73.479	10.174
Banana	689.451	1e+9	153	0.758	152.242	130.830	0.675	135.495	137.255
Orange	4,755.200	1e+9	135	4.280	130.720	81.310	2.011	61.439	132.667
Peach	9.986	1e+9	115	0.010	114.990	37.830	0.004	44.846	50.134
Pear	203.395	1e+9	134	0.211	133.789	76.610	0.100	62.881	87.872
								Mean absolute percentage error	98.860

Table 12 Mobile testing uncalibrated, mass and volume estimation

	Apple, kcal	Banana, kcal	Orange, kcal	Peach, kcal	Pear, kcal	MAPE
MyFitnessPal	81	148	63	53	63	7.155
Nutra Check +	58	78	36	35	50	20.219
SnapCalorie	111	114	66	69	106	38.138

Table 13 Calculated calories from existing applications

	Apple	Banana	Orange	Peach	Pear	MAPE
Lab	16.417%	17.765%	27.551%	17.688%	25.655%	21.015%
Phone*	66.849%	63.002%	57.330%	20.346%	51.977%	51.901%
MyFitnessPal	8.173%	8.688%	0.709%	18.172%	0.032%	7.155%
Nutra Check	22.543%	42.719%	43.262%	21.962%	20.610%	30.219%
SnapCalorie	48.237%	16.281%	4.019%	53.846%	68.307%	38.138%

Table 14 Numerical comparison in percentage error between pipeline and commercial alternatives

Image Recognition

```
#! Train architecture with the best architecture
# Split into train/test datasets using all of the pictures
train_df,test_df = train_test_split(df, test_size=0.1, random_state=0)
# Create the generator
train_generator,test_generator,train_images,val_images,test_images=create_gen()
# Create and train the model
model = get_model(tf.keras.applications.DenseNet201)
history = model.fit(train_images, validation_data=val_images, epochs=5, callbacks=[tf.keras.callbacks.BackupAndRestore(backup_dir="/tmp/backup",
                save_freq='epoch', delete_checkpoint=True),tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=1, restore_best_weights=True)])
model.save("C:.../Saved Models/foodRecognitionClassifier_DenseNet201.keras")
```

Figure 28 Code snippet of TensorFlow transfer learning on image recognition

```
Epoch 1/5
1786/1786 [==============================] - 699s 387ms/step - loss: 0.0767 - accuracy: 0.9787 - val_loss: 0.0081 - val_accuracy: 0.9972
Epoch 2/5
1786/1786 [==============================] - 327s 183ms/step - loss: 0.0177 - accuracy: 0.9942 - val_loss: 0.0102 - val_accuracy: 0.9957
221/221 [==============================] - 79s 344ms/step
# Accuracy on the test set: 99.63%
```

Figure 29 Image recognition console of training results

```python
# Load the paths of training image dataset - to read classification number
images = []
directory = r"C:...\\Fruit Image DB"
for f in os.listdir(directory):···

# Create a dataframe with the paths and the label for each fruit
df = pd.DataFrame(images, columns = ["fruit", "path"])

# Shuffle the dataset
from sklearn.utils import shuffle
df = shuffle(df, random_state = 0)
df = df.reset_index(drop=True)

# Assign to each fruit a specific number
fruit_names = sorted(df.fruit.unique())
mapper_fruit_names = dict(zip(fruit_names, [t for t in range(len(fruit_names))]))
df["label"] = df["fruit"].map(mapper_fruit_names)


##* Loading images to be tested ##
#need to find a way to dynamicaly send the file location to
image="C:...\\image.jpg"
unknownFruit = plt.imread(image)
unknownFruit=(cv2.resize(unknownFruit, (224,224)))
unknownFruit=unknownFruit.reshape(1,224,224,3)

##* Loading Pre-trained Saved Model to quickly identify the fruit##
loadingAModel = tf.keras.models.load_model('C:.../Saved Models/foodRecognitionClassifier_DenseNet201.keras')

# Predict the label of the test_images
predict = loadingAModel.predict(unknownFruit)
predictedClass = np.argmax(predict,axis=1)
print("Predicted Class: ",predictedClass)

strippedText = str(predictedClass).replace('[','').replace(']','')
value=list(mapper_fruit_names.keys()) [list(mapper_fruit_names.values()).index(int(strippedText))]
print("Predicted Value: ", value)
confidence_score = tf.math.reduce_max(predict, axis=1)
print("Confidence score: ", confidence_score)
```

Figure 30 Code snippet of image recognition predicting the image class and accuracy

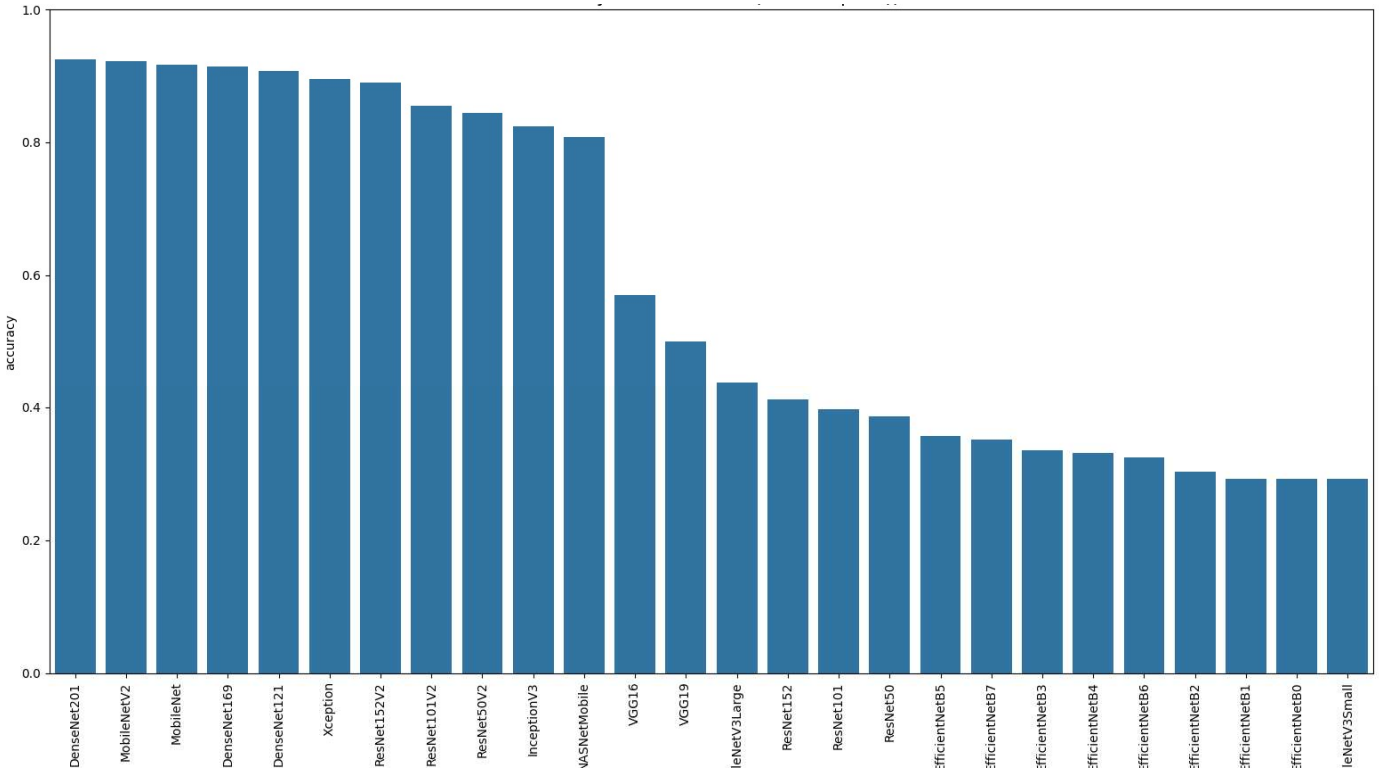


Figure 31 Accuracy of ML model on test set after 1 epoch

Object Detection

```python
import cv2 as cv
import numpy
from ultralytics import YOLO
import os
os.environ['KMP_DUPLICATE_LIB_OK']='TRUE'

# Load a pretrained model
model = YOLO("yolo11n.pt")

results = model.train(data="C:.../open-images-v7-COCO-v05\dataset.yaml", epochs=100, imgsz=640,workers=0, batch=-1, patience=5, optimizer="auto",
                      lr0= 0.00269, lrf= 0.00288,momentum= 0.73375, weight_decay= 0.00015, warmup_epochs= 1.22935, warmup_momentum= 0.1525,
                      box= 18.27875, cls= 1.32899, dfl= 0.56016, hsv_h= 0.01148, hsv_s= 0.53554,hsv_v= 0.13636,
                      degrees= 0.0, translate= 0.12431, scale= 0.07643, perspective= 0.0, flipud= 0.0, fliplr= 0.08631,
                      mosaic= 0.42551, mixup= 0.0,copy_paste= 0.0)

model.save("yolo11-Version7.pt")
```

Figure 32 Code snippet of transfer learning a YOLO 11 model with hyperparameters

```python
import cv2 as cv
import numpy
from ultralytics import YOLO
import os
os.environ['KMP_DUPLICATE_LIB_OK']='TRUE'

import torch
if torch.cuda.is_available():
    print("gpu detected")
else:
    print("gpu not detected!")
    exit()

# Load custom trained model
model = YOLO("yolo11-Version7.pt")

imageRaw = cv.imread("image.jpg")# Load image to be read
#cv.imshow("Image", imageRaw), cv.waitKey(0)# output original image # Wait for a key press

# Run YOLO model on the captured frame and store the results
results = model.predict(imageRaw, device="cpu") # dependant on the number of images provided, imageRaw == index[0]
#results = model.predict(source=imageRaw, conf=0.5)
annotated_image = results[0].plot()

predictedClass = results[0].boxes.cls.numpy()
predictionConfidance = results[0].boxes.conf.numpy()

for index in range(predictedClass.size):
    className=model.names.get(int(predictedClass[index]))
    strippedClassConfidance = str(predictionConfidance[index]).replace('[','').replace(']','').replace(' ', '') # strips & sanatises text
    print("Item No. ", index, "\nPredicted Class: ",className,"\nPrediction Confidance: ",strippedClassConfidance)
```

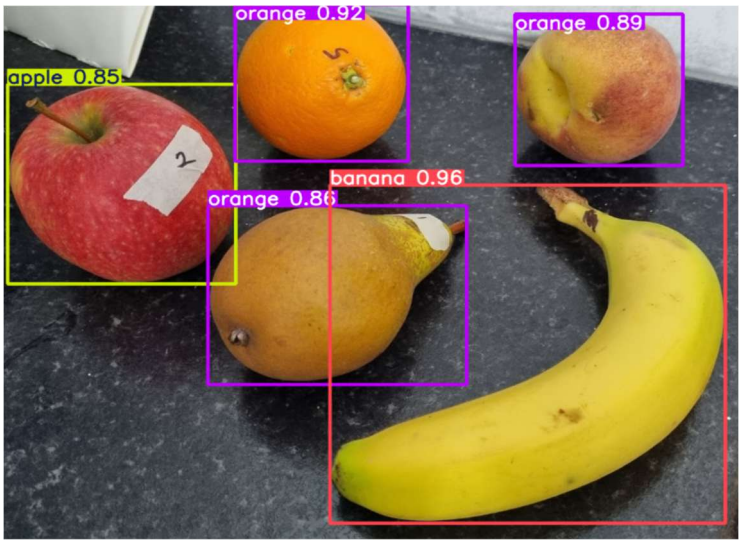
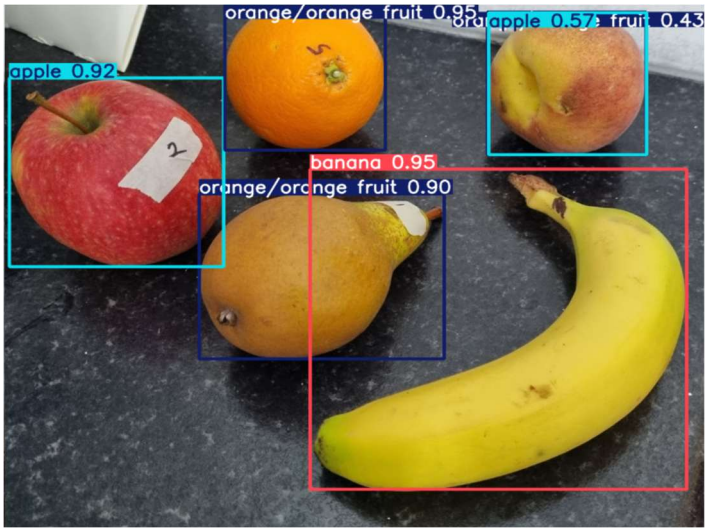
Figure 33 Code snippet of YOLO11 object detection singles images' classes and accuracy

foodDensity.csv > data

You, 2 months ago | 1 author (You)

```
fruit,rioeness,density,units,source
apple,ripe,950,kg m^-3,https://typeset.io/pdf/physical-and-chemical-properties-of-three-late-ripening-4pk8nflcnu.pdf
banana,ripe,1100,kg m^-3,http://researchjournal.co.in/upload/assignments/3_251-255.pdf
orange,ripe,900,kg m^-3,https://www.sciencedirect.com/science/article/abs/pii/S026087740400189X
peach,ripe,998,kg m^-3,https://www.sciencedirect.com/science/article/abs/pii/S0260877414002489
pear,ripe,1036,kg m^-3,https://www.tandfonline.com/doi/full/10.1081/JFP-120025392      You, 2 months ago • LVIS training
```

Figure 34 Fruit density database

Dataset	Result Image	Output Console
Default-COCO		0: 544x640 1 banana, 1 apple, 3 oranges, 93.5ms Speed: 5.0ms preprocess, 93.5ms inference, 2.0ms postprocess per image at shape (1, 3, 544, 640) Item No. 0 Predicted Class: banana Prediction Confidence: 0.95874745 Item No. 1 Predicted Class: orange Prediction Confidence: 0.9204238 Item No. 2 Predicted Class: orange Prediction Confidence: 0.89302474 Item No. 3 Predicted Class: orange Prediction Confidence: 0.8600857 Item No. 4 Predicted Class: apple Prediction Confidence: 0.85022515
LVIS		Speed: 6.0ms preprocess, 95.0ms inference, 4.0ms postprocess per image at shape (1, 3, 544, 640) Item No. 0 Predicted Class: banana Prediction Confidence: 0.9544522 Item No. 1 Predicted Class: orange/orange fruit Prediction Confidence: 0.9493486 Item No. 2 Predicted Class: apple Prediction Confidence: 0.91742504 Item No. 3 Predicted Class: orange/orange fruit Prediction Confidence: 0.8956704 Item No. 4 Predicted Class: apple Prediction Confidence: 0.5676047 Item No. 5 Predicted Class: orange/orange fruit Prediction Confidence: 0.4303231

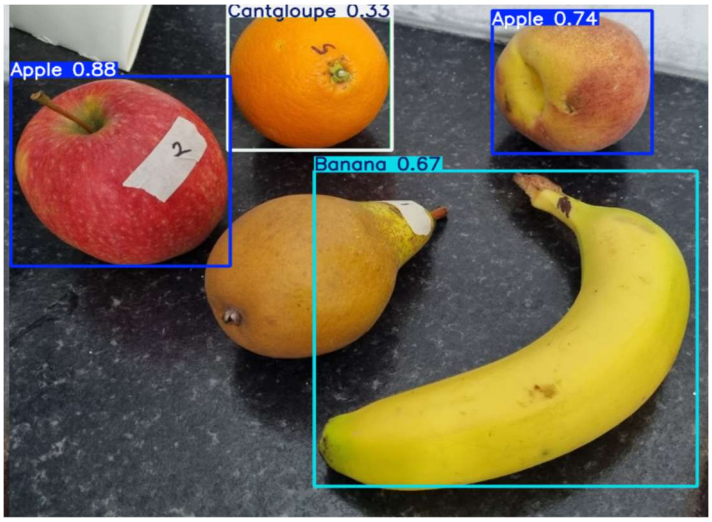
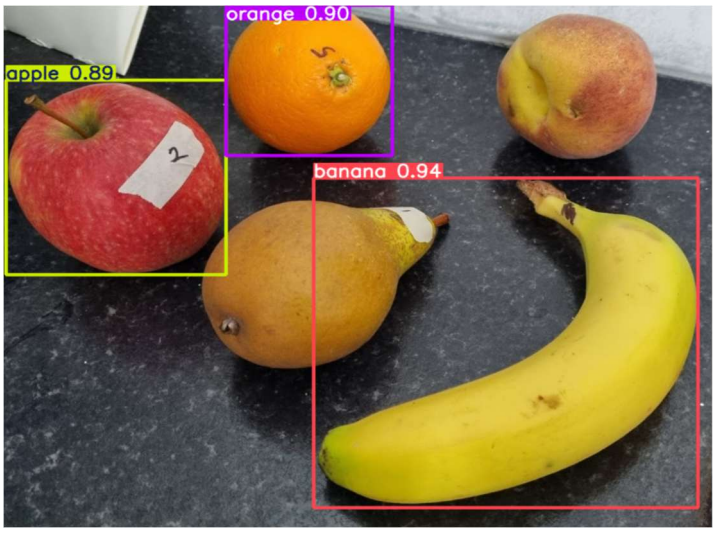
Open image v5		0: 1056x1280 2 Apples, 1 Banana, 1 Cantaloupe, 1 Orange, 217.6ms Speed: 14.0ms preprocess, 217.6ms inference, 2.0ms postprocess per image at shape (1, 3, 1056, 1280) Item No. 0 Predicted Class: Apple Prediction Confidence: 0.88096577 Item No. 1 Predicted Class: Apple Prediction Confidence: 0.73924404 Item No. 2 Predicted Class: Banana Prediction Confidence: 0.673159 Item No. 3 Predicted Class: Cantaloupe Prediction Confidence: 0.32991856 Item No. 4 Predicted Class: Orange Prediction Confidence: 0.25744906
Yolov8n		0: 544x640 1 banana, 1 apple, 1 orange, 108.9ms Speed: 5.6ms preprocess, 108.9ms inference, 2.0ms postprocess per image at shape (1, 3, 544, 640) Item No. 0 Predicted Class: banana Prediction Confidence: 0.9408983 Item No. 1 Predicted Class: orange Prediction Confidence: 0.8991952 Item No. 2 Predicted Class: apple Prediction Confidence: 0.8888753

Table 15 Comparison between object detection datasets in class accuracy.

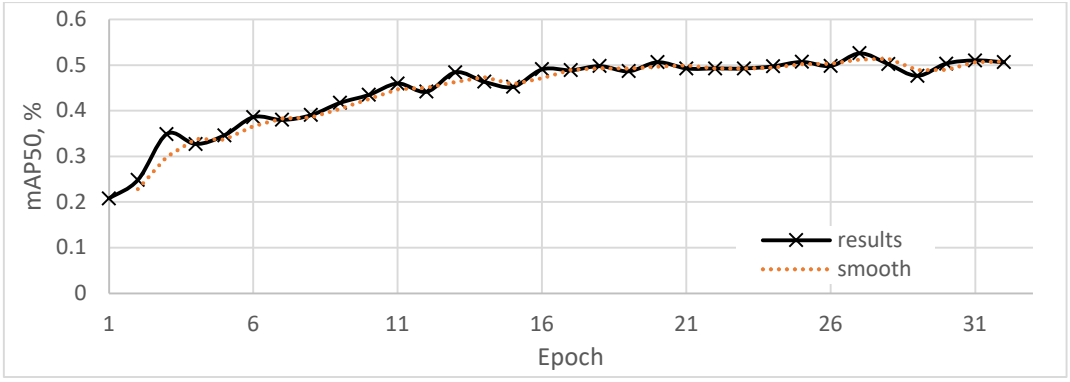


Figure 35 YOLO11n training mAP50 model accuracy with LVIS dataset.