Radboud University

Master Thesis

Mobile 3D Computer Vision:
Introducing a portable system for potato size grading

Author:
Remco Runge
Department of Artificial Intelligence
Radboud University Nijmegen

Internal supervisor:
dr. L.G. Vuurpijl
Department of Artificial Intelligence
Radboud University Nijmegen

External supervisor:
ir. R. van Tilborg
Smart Technologies
Ordina Nieuwegein

December 10, 2014

Abstract

Computer vision has gained an important place in the agricultural sector. It is used to measure and grade agricultural products such as potatoes, apples and oranges. Most of these systems rely upon dedicated and expensive computer vision setups. Within this thesis, an inexpensive and portable grading system for potato tubers is presented.

The presented system utilises 2D and 3D computer vision techniques to estimate the length, width and square mesh size of a potato tuber. The rapid adaptation of smartphones and smartglasses (such as the Google Glass) has made it possible to capture and process images at relatively low costs. Within this thesis, the use of a smartphone and smartglass as a inexpensive system for mobile potato grading is investigated.

The results of this thesis show the potential of inexpensive potato grading based on 2D and 3D computer vision for images captured with mobile devices. The future of agricultural grading can be mobile!

Contents

Acknowledgements

First of all, I would like to thank my supervisors Louis Vuurpijl and Richard van Tilborg, for their guidance and encouragement during my research and my writing. Furthermore, I would like to thank all my colleagues at Ordina SMART technologies for their help and advice. I thank my family and friends for their support, feedback and for helping me structuring my mind.

Chapter 1

Introduction

The world population is ever growing. With every new child born, there is a new mouth to feed. This puts a high pressure on the agricultural sector to deliver high quality and affordable products. Automation is therefore key to reduce labour costs and improve quality. Human operators are gradually being replaced by automated systems which are in most cases faster and more precise (Narendra and Hareesha, 2010).

To answer this need, computer vision has gained an important role in the agricultural sector to automate processes. Computer vision is a technique which is used to analyse images of real scenes by computers in order to derive information which in turn can be used to control machines or processes. The core of computer vision is related to the field of image analysis and image processing which is used to quantify and classify images and objects of interest within images (Sun, 2004).

Within the agricultural sector the use of computer vision is mainly focussed on automating the quality inspection, classification and evaluation of a wide range of agricultural products. Agricultural products need to be sorted and graded for commercial and production purposes. Traditional, grading and inspecting agricultural products is done by human operators. This manual process is often tedious, inaccurate, time-consuming and inconsistent. Current research aims at automating this labour intensive process (Narendra and Hareesha, 2010)

Systems have been developed to sort, inspect and grade a wide variety of agricultural product such as apples (Li et al., 2002), tomatoes (Jahns et al., 2001), olives (Riquelme et al., 2008), grains (Paliwal et al., 2003), potatoes (Rios-Cabrera, 2008) etc. In Chapter 2 a more in-depth overview of the use of computer vision in the agricultural sector will be given.

1.1 Project introduction

The computer vision systems discussed above all rely upon relatively expensive camera set-ups. This makes them infeasible to use for automating smaller scale agricultural processes. One of such problems comes to light when farmers need to determine the growth of their potatoes. During this process only a small section of the potato field is harvested after which the size of each potato is measured in order to determine whether or not the rest of the field is ready for harvest.

Currently, this process is completely performed by hand. Each individual potato needs to be measured with the help of a square mesh size measuring tool (Fig. 1.1). The square mesh size is one of the most common criteria for size grading around the world. The square mesh size of a potato is defined as the smallest square aperture through which a potato can be pushed lengthwise without effort.

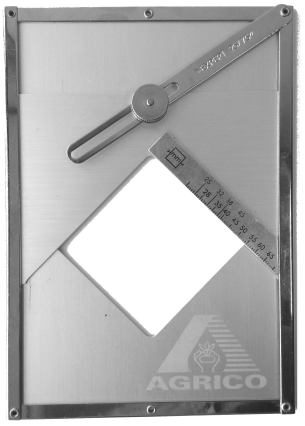


FIGURE 1.1: Tool to measure the potato square mesh size.

The sampling process is done at a small scale of around 180 potatoes at a time. Although the sampling and measuring process is quite labour intensive, the small scale does not justify the high costs of a dedicated computer vision set-up.

With the rapid development of camera equipped mobile devices in the last couple of year, it has become possible to create mobile and relatively cheap computer vision applications. Devices such as smartphones, tablets and smart glasses[1] are equipped with increasingly more computational power and better cameras. They have become computationally powerful enough to handle basic computer vision tasks such as facial recognition (Cheng and Wang, 2011).

[1] Smartglasses are wearable computers in the form of glasses with a display mounted to the frame. Often these smartglasses are also equipped with a camera. The Google Glass is a well known example of a smartglass.

Thanks to the connectivity of these devices it is also possible to offload the more computational intensive computer vision tasks to remote resources (Kemp et al., 2012). Therefore, the use of mobile devices equipped with a camera could, due to their mobility and relatively low costs, be a viable platform for mobile agricultural product grading on a small scale.

We therefore propose a low cost computer vision system for potato square mesh size determination based on images captured with a smartphone or smartglass by measuring the minor (height) and intermediate (width) axis of a potato tuber. Figure 1.2 gives a schematic overview of the dimensions of a potato tuber.

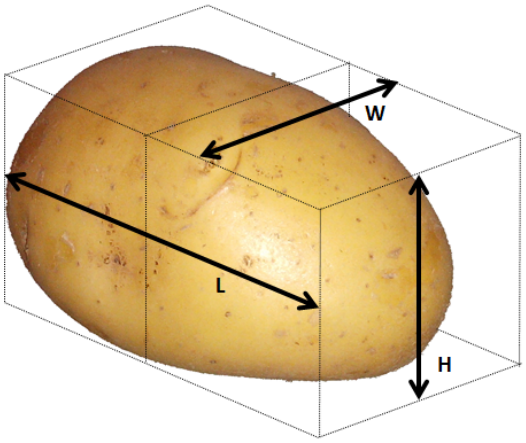


FIGURE 1.2: Schematic overview of the dimensions of a potato: the major axis/length (L), intermediate axis/width (W) and the minor axis/height (H).

Within this thesis, the implementation of such a system will be described. Furthermore, the research questions described in the next section will be answered.

1.2 Research Questions

- "What is the viability of a computer vision potato grading system for mobile devices?"

To assess the viability, the following sub questions should be answered:

- "How accurate can the system measure the length of the minor axis (height) of a potato tuber?"
- "How accurate can the system measure the length of the intermediate axis (width) of a potato tuber?"

- "How accurate can the system derive the mesh size of a potato based on the measured length of the intermediate and minor axis of the potato tuber?"
- "How well does the system handle measuring multiple potatoes at the same time?"

Private discussions with relevant experts from the potato industry showed that a mean absolute error of 3 millimetre for the square mesh size determined by the computer vision system compared to measurements with the square mesh size measuring tool (Fig. 1.1), would be an acceptable result. Therefore the system is considered viable when the mean absolute error is below 3 millimetres.

1.3 Review of the current field of computer vision

The field of computer vision is rapidly developing. The number of available algorithms and methods for computer vision keeps increasing. These algorithms and methods have been implemented in a wide variety of computer vision libraries, toolboxes and frameworks. To investigate which of these software packages would be the best basis for creating the proposed system, a review of the current field of computer libraries, toolboxes and frameworks was created preliminary to this study. This review can be found in Appendix A.

1.4 Organization of this thesis

This thesis is organized in six chapters. Within this chapter, the general introduction was outlined, and the project and research questions were introduced. In Chapter 2, an overview of the research context is given. The methods, implementation and experimental setup of the system are described in Chapter 3. In Chapter 4 the results of this study are presented, which are discussed in Chapter 5. This thesis is organized in six chapters. Within this chapter, the general introduction was outlined, and the project and research questions were introduced. In Chapter 2, an overview of the research context is given. The methods, implementation and experimental setup of the system are described in Chapter 3. In Chapter 4 the results of this study are presented, which are discussed in Chapter 5.

Chapter 2

Research Context

In the 1960's some Artificial Intelligence and Robotics researcher saw the 'visual input' problem as a relatively easy step along the path of solving complex problems such as higher-level reasoning and planning. This is illustrated by a famous story from 1966 in which Marvin Minsky at MIT asked his undergraduate student Gerald Jay Sussman to "spend the summer linking a camera to a computer and getting the computer to describe what it saw" (Boden, 2006). As we now know, learning a computer to describe what it sees is not just a summer project, but it is a complete field of research.

As noted in the introduction, agricultural computer vision has gained an important place in the field of computer vision. Within this chapter the current research in the field of agricultural computer vision will be described.

2.1 Computer vision in the agricultural sector

2.1.1 Potatoes

Potatoes come in all kinds of different shapes and sizes. Different markets demand differently shaped potatoes. It is therefore necessary to grade the potatoes into different uniform classes depending on the market.

Tao et al. (1995) developed a Fourier based shape separation method using computer vision for automatic grading of green and good potatoes. In their work, they defined a separator based on the harmonics of the Fourier transform. The accuracy of their system was 89% for 120 potato samples. This result was in line with manual grading performed by experts and farmers.

Heinemann et al. (1996) built a prototype inspection station based on the United States Department of Agriculture (USDA) inspection standards for potato grading. Potatoes were individual photographed in the systems image chamber after which shape and size were estimated. The system was able to get a maximum classification rate of 98%.

A high-speed computer vision system capable of classifying 50 potato images per second has been presented by Zhou et al. (1998). The system evaluated weight, cross-sectional diameter, colour and shape of three different cultivars of potatoes. An ellipse was fitted to the image of a potato as a shape descriptor. Colour thresholding in the HSV (Hue-Saturation-Value) colour space was performed to detect green colour defects. The system was able to achieve a average success rate of 91.2% for weight inspection and 88.7% for diameter inspection. Furthermore, it was able to achieve a colour inspection success rate of 78.0% and a shape inspection success rate of 85.5%. Overall the system had an average success rate of 86.5

More recently Rios-Cabrera et al. (2008) employed Artificial Neural Networks (ANN) to determine the quality of potatoes by evaluating physical properties and detecting misshapen potatoes. Three different connectionist models (Backpropagation, Perceptron and FuzzyARTMAP) were evaluated on speed an stability for classifying extracted properties. FuzzyARTMAP outperformed the other models on stability and convergence speed with values lower than 1ms per pattern. The fast processing algorithm makes the methodology suitable for quality control in production lines.

Two different types of potato inspection approaches were evaluated by Jin et al. (2009). Adaptive Intensity Interception and Fixed Intensity Intersection were compared for tubers with defects using Otsu segmentation in combination with morphological operators. The results showed that the latter method was more effective for tuber defect inspection.

Al-Mallahi et al. (2010) developed a computer vision to automatically detect potato tubers and lumps of earth and clay (clods). The ultraviolet reflectance of tubers compared to their background which includes pieces of clods, was used for the detection. The tubers were segmented by estimating their size by calculating their maximum size and width. A total of 1171 video frames which included 2233 tubers and 1457 clods were segmented. The system was able to successfully detect 98.79% of the tubers and 98.28% of the clods.

Hasankhani and Navid (2012) created a computer vision system for grading potatoes into three categories based on size. The size of the potatoes was determined by thresholding the image in the HSV (Hue-Saturation-Value) colour space, after which the boundaries were extracted. A total number of 110 potatoes were sorted by the system with an average precision of 96.823%.

The systems described in this section are all systems that require dedicated and stationary camera set-ups. This makes these systems relatively expensive. Furthermore, none of these systems incorporates the 3D shape of the potato.

2.1.2 Apples

Paulus and Schrevens (1999) created an algorithm to determine the phenotypes of an apple by characterizing objectively the shape of the apple with the help of Fourier expansion. The dimensionality of the edge points of the image of an apple were reduced to a set of 24 Fouriers coefficients. Principle component analysis on the set of Fourier coefficients was used to get two shape variables which were used to measure accurately the apple profiles described by a subjective descriptor list. Within this research, they determined the need for at least four images of a randomly chosen apple in order to quantify its average shape. The algorithm was successfully able to distinguishing the shape of different apple cultivars.

Research by Paulus et al. (1997) gave insight into the way in which external product features can affect the human perception of quality. 'Tree-based modelling' was used to simulate apple quality classification using objective measurement of external properties. It was found that the characteristics which influence the classification differ according to the variety of the apple. Furthermore, it was found that humans were inconsistent in their quality estimation. According to this study, the inconsistency is influenced by the amount and complexity of the product features. The higher the amount and complexity of the product features became, the error of human classification increased.

Apple defects are an important factor when grading apples. Several studies into analysing with the help of computer vision have therefore been performed. Leemans et al. (1998) used computer vision to find and segment defects on 'Golden Delicious' apples. Defects were segmented by comparing the Mahalanobis distance of each pixel of the image of an apple to a global model of healthy fruits.This method turned out to be effective in detecting a variation of defects such as bruises, russet, scab, fungi or wounds.

Yang (1996) investigated the feasibility of using computer vision for automatic grading and coring of apples by detecting stems and calyxes. The proposed method uses a back propagation neural network to classify each patch as either stem/calyx or patch like blemish. On a sample of 69 Golden Delicious and 55 Granny smith samples, this method was able to achieve an overall accuracy of 95%.

Xiao-bo et al. (2010) based their system on three cameras to identify apple stem-ends and calyxes from defects. By rotating the apple in front of the camera's a total of

9 pictures were taken (three by each camera) to capture the whole apple. The apple image was segmented from its background by multi-threshold methods, after which the the defects, including the stem-ends and calyxes were segmented as Regions Of Interest (ROI's). Based on the fact that stem-ends and calyxes can not appear in the same picture, an apple was marked as defected if at least two ROI's were visible in the image.

2.1.3 Other agricultural products

Watermelons In research by Koc (2007), the volume of a watermelon was estimated with the help of computer vision and ellipsoid approximation. The resulting estimated volume by computer vision, did not significantly differ from the volume measured by water displacement.

Oranges An image processing algorithm to determine the volume and surface area of oranges was created by Khojastehnazhand et al. (2009). The created system made use of two cameras and an appropriate lighting system. By placing the cameras at a right angle to each other, a perpendicular view of the orange was created. The algorithm segmented the background and divided the image into a number of frustums of right elliptical cone. The volume and surface area of each of the frustums were then computed by the segmentation method. By summing all elementary frustum, the total volume and surface area of the orange was approximated.

Strawberries Liming and Yanchao (2010) developed a computer vision to grade strawberries based on shape, size and colour. The strawberries were segmented with the Otsu method. Line sequences were then extracted from the strawberries contours to express the strawberries shape, after which the shape parameters were clustered with k-means clustering. The system was able to detect the strawberry size with a detection error below 5%. Grading based on the colour was done with an accuracy of 88.8% and the shape classification accuracy was above 90%.

2.2 Square mesh size

The size of a potato is one of the most important grade attributes in the potato processing industry. In most countries the square mesh size is accepted as a standard sizing criteria for potato tubers (Struik et al., 1990). The square mesh size is defined as the smallest square aperture through which a potato tuber can be pushed lengthwise without any pressure and without damaging the tuber. Potato grades are often expressed

by the lower and upper size of a square aperture. A grading of 35/40mm would then entail that the tuber will not pass a 35mm sized aperture, but will pass a 40mm sized aperture.

During manual grading, the smallest square mesh size is determined by using trial and error to fit potatoes through a square mesh size tool such as displayed in Figure 1.1. This sampling method is based upon the tubers largest transverse cross-section to be the critical potato characteristic (De Koning et al., 1994). To be more precise, when a tuber is orientated with its largest cross-sectional dimension on line with the diagonal of the square aperture, the square mesh size is then defined as the length of the square that exactly circumscribes the largest transverse cross-section of a tuber.

Research by De Koning et al. (1994) introduced a method to derive the square mesh size of a potato from the length of its minor-axis (height) and the length of its intermediate axis (width) (Equation 2.1).

$$S = \sqrt{\frac{W^2 + H^2}{2}} \tag{2.1}$$

In which:

S = Square mesh size (mm).

L = Length of the major axis of the tuber (mm).

W = Width, the next largest dimension perpendicular to L (intermediate axis of the tuber).

H = Height, the next largest dimension perpendicular to both L and W (minor axis of the tuber).

Chapter 3

Methods

Within this chapter, the methods used to create and test the mobile computer vision system for potato square mesh size detection are discussed.

3.1 Materials

The proposed system to measure the the square mesh size of a potato tuber consists of a mobile image capturing device, a marker board for camera position tracking and a central server for processing the images. Furthermore, 10kg potato tubers from the 'Melody' variety were used to test the system.

Image capturing devices Within this thesis both a Xiaomi RedMi smartphone and a Google Glass smartglass were used as image capturing devices. The Xiaomi RedMi comes equipped with a 8 megapixel camera, while the Google Glass uses a 5 megapixel camera.

Marker board A special board was designed as a surface on which the potato tubers were placed (Fig. 3.1). The board consisted of a sheet of A3 sized paper. At the border of the board, highly reliable fiducial markers were placed (Garrido-Jurado et al., 2014). Based upon these markers, the location of the board could be detected with high precision. Furthermore, the markers made it possible to track the relative position of board in relation to the position of the camera. Tracking of the cameras relative position to the board was necessary in order to create a three dimensional image of the potato tuber.

Within the boarder of highly reliable fiducial markers, a black background was chosen to help the segmentation of the potatoes from the background.

FIGURE 3.1: Board with highly reliable fiducial markers based on the ArUco library (Garrido-Jurado et al., 2014).

Central server A HP MobileWorkstation EliteBook 8740w was used as a central server to perform the computer vision. This notebook was equipped with a Intel i7 Q840 1.87Ghz quad-core processor and a total of 16GB internal RAM. Windows 7 was used as the operating system.

3.2 Computer vision system design

To derive the potato square mesh size with the help of the equation created by De Koning et al. (1994) both the width of the potato (length of the intermediate axis) and the height (length of the minor axis) of the potato have to be known (Section 2.2). To estimate the width and height of the potato tuber with computer vision, two different approaches were taken. The width of the potato was measured by using 2D computer vision. While the height of the potato was measured by using 3D computer vision.

The system was build in C++ and makes use of three software libraries. All three libraries are available under the BSD license, which makes the free to use for both for research as well as commercial use.

OpenCV The OpenCV library (Bradski, 2000) is one of the most widely used computer vision libraries available. It was originally developed by Intel and is now supported by Willow Garage and Itseez. It houses a wide variety of computer vision algorithms as can be seen in the table in section A.6.4. It was chosen as the basis of the potato square mesh size detection based upon the review of computer vision libraries, frameworks and toolboxes A.

ArUco Both the creation as well as the detection of the fiducial markers on the marker board were performed with the ArUco software library. Markers created with this system have high inter-marker distances and lower false negative rates compared to other fiducial marker systems (Garrido-Jurado et al., 2014). Based upon these markers, the outline of the board could be detected with hight precision. Furthermore, the relative position to the board in relation to the camera could be tracked.

Point Cloud Library The Point Cloud Library is a well used library to visualize three dimensional data into a point cloud (Rusu and Cousins, 2011). Within this thesis it is used to visualize the three dimensional point cloud of the potatoes.

3.2.1 Camera resectioning

Camera resectioning[1] is the process of finding the perspective transformation characteristic of a camera that produced a given image. Determining these characteristics is key for accurate computer vision based measurements and for the creation of a 3D model based on 2D images with multi-view stereo computer vision.

The parameters for the perspective projection can be split into *intrinsic parameters* and *extrinsic parameters*. The intrinsic parameters describe the optical apparatus, the actual projection mechanism and the distortion, while the extrinsic parameters describe the camera position and view direction. In the next section, finding the intrinsic parameters is discussed. Finding the extrinsic parameters is discussed in Section 3.4.3.

3.2.1.1 Intrinsic parameters

The projective transformation are defined by the intrinsic parameters in matrix K.

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

In which f_x and f_y are the focal lengths in pixel unit, c_x and c_y are the $x-$ and $y-$coordinates of the principal point in pixel unit (the intercept point of the optical axis and the projective plane) as depicted in Figure 3.3.

[1] Camera resectioning is also often refereed to with the term camera calibration. However the term camera calibration can also refer to the mapping of colours between two images. Due to this ambiguity, within this thesis, only the term camera resectioning will be used from now on.)

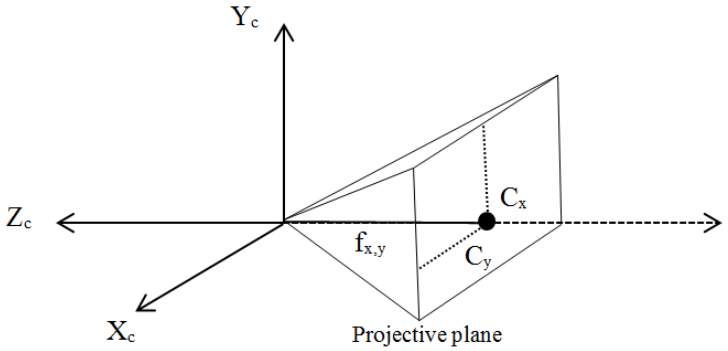


FIGURE 3.2: Intrinsic parameters

The K matrix can be used to determine the projection coordinate (u, v) of an arbitrary 3D point M_c (expressed in the camera coordinate system):

$$\begin{bmatrix} x & y & w \end{bmatrix} = K * M_c = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} M_{X_c} \\ M_{Y_c} \\ M_{Z_c} \end{bmatrix} \tag{3.2}$$

and

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = 1/w * \begin{bmatrix} x \\ y \\ w \end{bmatrix} \tag{3.3}$$

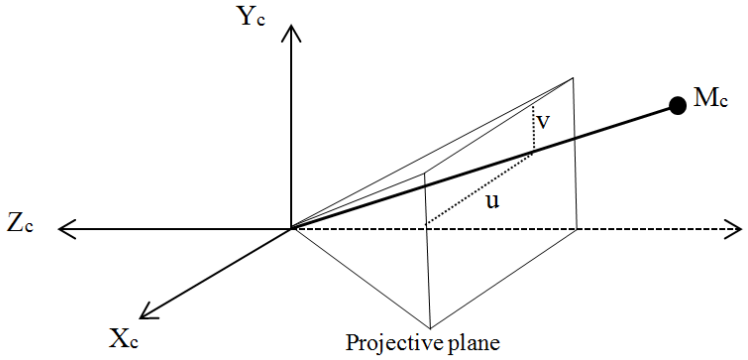


FIGURE 3.3: Projective transformation

Since smartphone and smartglasses cameras are not perfect pinhole camera models, the distortion caused by the (often plastic) lenses has also to be taken into account.

The images taken with are affected by both radial and tangential distortion. Radial distortion will make straight lines appear curved. Tangential distortion makes some part of the image look nearer then expected and is the result of image lenses not being

exactly aligned parallel to the image plane. These distortions vary between different cameras. A camera's distortion can be expressed by a one row, five column vector:

$$Distortion_{coefficients} = (k_1, k_2, t_1, t_2, k_3) \tag{3.4}$$

in which k_1, k_2, k_3 are the radial distortion coefficients and t_1, t_2 are the tangential distortion coefficients.

Calculation of the radial distortion coefficients uses the following equation:

$$x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{3.5}$$
$$y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{3.6}$$

The tangential distortion coefficients are calculated as follows:

$$x_{corrected} = x + [2p_1 xy + p_2(r^2 + 2x^2)] \tag{3.7}$$
$$y_{corrected} = y + [p_1(r^2 + 2y^2) + 2p_y xy] \tag{3.8}$$

The inbuilt camera resectioning methods of the OpenCV library Bradski (2000) were used to determine the intrinsic parameters. These methods are based on the work by Zhang (2000).

3.3 2D detection of the tubers width

Within this section, the techniques used to create an estimation of the width of a potato based on 2D computer vision are described. Fig. 3.4 describes the computer vision pipeline used to estimate the width of a potato tuber based on an image taken perpendicular to the surface of the board.

Three pre-processing steps are taken (Sec. 3.3.1). First, the marker board is used to detect the region of interest (Sec. 3.3.1.1), followed by Gaussian smoothing (Sec. 3.3.1.2) of the image to reduce the effect of noise. The last pre-processing step changes the colour space from RGB (red, green, blue) to the HSV (hue, saturation, value) colour space (Sec. 3.3.1.3). The feature detection phase (Sec. 3.3.2);consists of first thresholding the image to make the contours of each potato easy to detect (Sec. 3.3.2.1), followed by determining the minimum bounding box around those contours (Sec. 3.3.2.2). In the following sections, these steps are discussed in more detail.

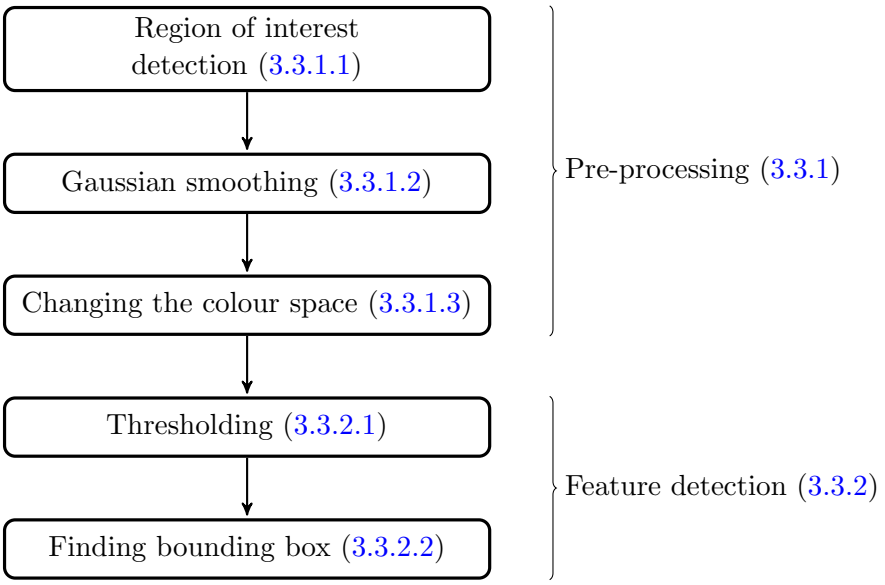


FIGURE 3.4: Computer vision pipeline for estimating the potato width on the basis of an image taken perpendicular to the board. The corresponding subsections are displayed between the brackets.

Input To estimate the width of a potato tuber, the system uses images taken perpendicular to the boards surface (Fig. 3.5.). The size of the input images was reduced to 1280*720 pixels to reduce the computational complexity while maintaining enough detail for the width estimation.

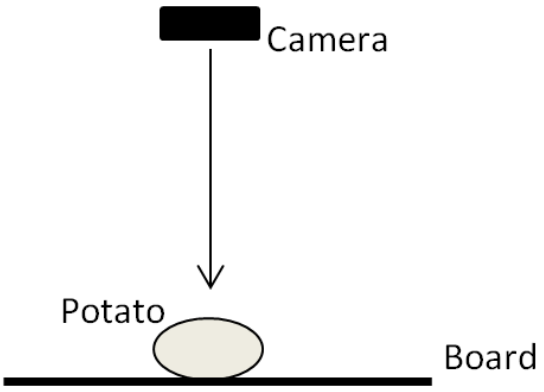


FIGURE 3.5: To estimate the width of the potato, a picture is taken perpendicular to the board.

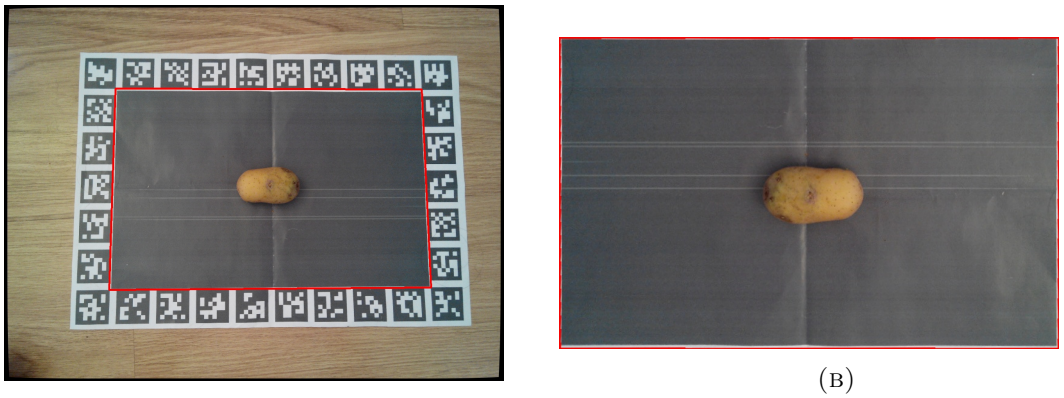
3.3.1 Pre-processing

The first step in most computer vision algorithms is the pre-processing step. The aim of this step is to remove unwanted variability in the image in order to make the rest of the computer vision tasks easier. Often, images taken with a cellphone camera can suffer

from random noise introduced by the camera sensor or by the compression used when saving the image.

3.3.1.1 Region of interest detection

The positions of the fiducial markers are used to determine the region of interest. Based on the coordinates of the recognized markers, the board is segmented from the rest of the image. After extraction of the region of interest, the region is rotated and transformed to a consisted rotation and rectangular form as can be seen in Figure 3.6.



(A) (B)

FIGURE 3.6: Rotated and transformed region of interest (B) from the original image (A)

3.3.1.2 Gaussian smoothing

To remove noise in the captured images, Gaussian smoothing was performed. Within this process, each pixel of the image is transformed with a Gaussian filter in order to smooth the image. The two dimensional Gaussian filter is described in formula 3.9. Within this formula, the x describes the distance from the origin of the horizontal axis, while the y describes the distance from the vertical axis, and the σ is the Gaussian distributions standard deviation.

The resulting Gaussian distribution from this filter can was then used to build a convolution matrix which was applied to the original image. This matrix was then used to set each pixels value to the weighted average of its self and its neighbours. Due to the bell shape nature of the Gaussian filter, the new value receives the heaviest weight from its original value, and lower weights from neighbouring pixel depending on the distance (the larger the distance, the smaller the weight).

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.9}$$

The result of Gaussian smoothing is an image in which smaller details such as noise are removed, while at the same time edges and boundaries of larger objects are preserved.

3.3.1.3 Colour space

The colour space of the image was transformed from the RGB (Red, Green and Blue) colour space to the HSV (Hue, Saturation and Value) colour space (Fig. 3.7). The HSV colour space relies upon the hue, saturation and brightness (value) of each pixel instead of their red, green and blue value. This colour space was developed to make it easier to handle illumination within the image. Within the HSV colour space, the potato can be easier segmented from its background as work by Zhou et al. (1998) shows.

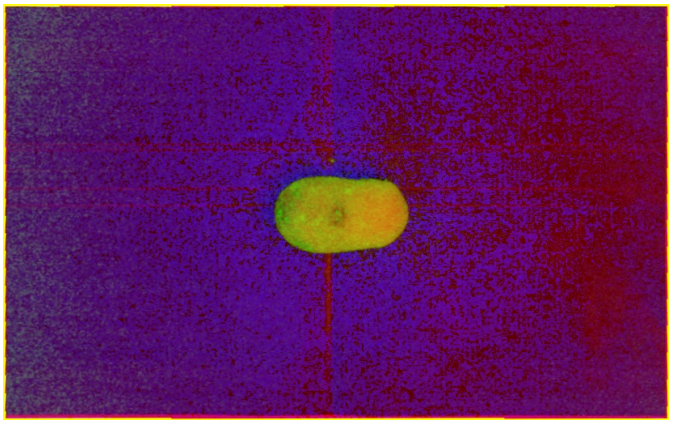


FIGURE 3.7: Image in the HSV colour space

3.3.2 Feature detection

3.3.2.1 Thresholding

To segment the potato tubers from their background, thresholding on the saturation channel of the HSV image was used (Fig. 3.8). Thresholding is a technique used to segment an object from its surrounding. In its most basic form pixels are categorized into one of two categories based on whether their value lies below or above a certain threshold.

Since this system has to be able to handle changes in lighting conditions, the well-established Otsu method (Otsu, 1975) was used to automatically determine the threshold which segments the potato tuber from its background. This method tries to segment the image

in to two clusters by finding the threshold that minimizes the weighted within-class variance in the histogram. The within-class can be defined as the weighted sum of variance of each cluster:

$$\sigma^2_{Within}(t) = w_1(t)\sigma^2_1 + w_2(t)\sigma^2_2(t) \tag{3.10}$$

in which the weights w_i are the probabilities of the two clusters separated by a threshold t and class variance σ^2_i

Work by Jin et al. (2009) showed that the using the Otsu method for thresholding is a viable way of segmenting a potato tuber from its background.

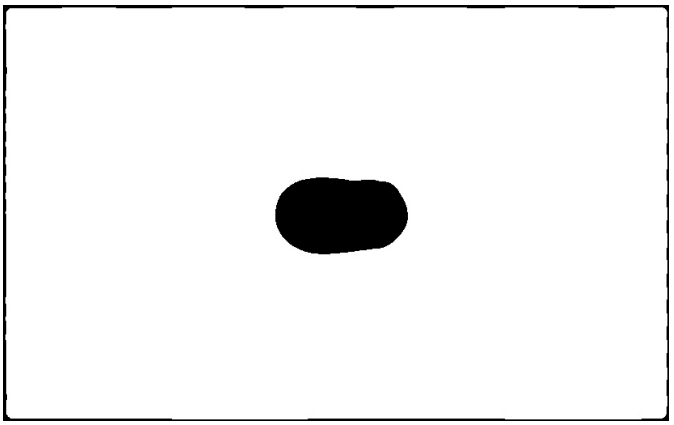


FIGURE 3.8: Thresholded image on the saturation channel.

3.3.2.2 Bounding box

To measure the width of each potato tuber, a bounding box was fitted around the contours of each group of pixels. This bounding box was rotated in all directions until the box with the minimum surface area was found (Fig. 3.9).

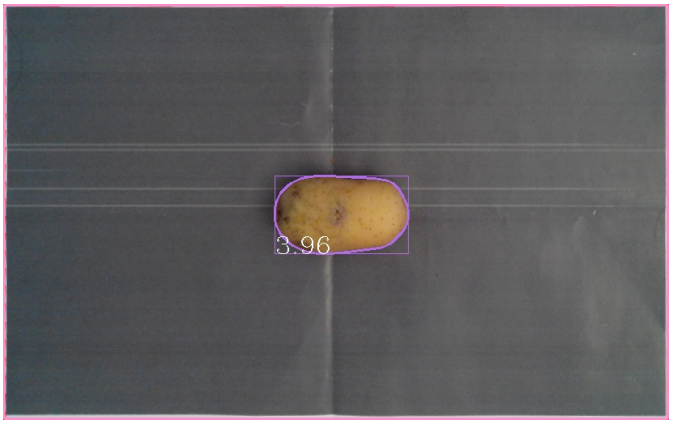


FIGURE 3.9: Finding the minimum bounding box (the number 3.96 depicts the width of the potato in centimetres).

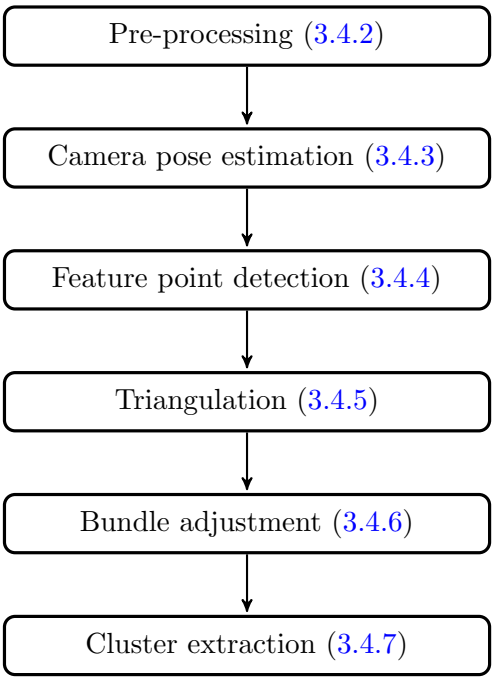


FIGURE 3.10: Computer vision pipeline for estimating the 3D shape of a potato on the basis of multiple 2D images. The corresponding subsections are displayed between the brackets.

3.4 3D detection of the tubers height

3.4.1 Multi-view stereo vision

In order to create a three dimensional model from multiple images, a technique called multi-view stereo vision can be utilized. Multi-view stereo vision makes use of multiple 2D images from the same object to derive a 3D view of the object. Figure 3.10 gives a schematic overview of the computer vision pipeline for creating a 3D model of a potato on the basis of multiple 2D images.

During the first step, the image is pre-processed to remove parts of the image that are of no use for the system to reduce the computational complexity (Sec. 3.4.2). The second step is estimating the camera pose relative to the marker board (Sec. 3.4.3). Third, feature points are detected and matched between images (Sec. 3.4.4), after which these feature points are triangulated based on the camera pose in step four (Sec. 3.4.5). During the fifth step, bundle adjustment is used to improve the triangulation of the found feature points (Sec. 3.4.6). Finally, the potato tubers are extracted from the resulting point cloud with the help of cluster extraction (Sec. 3.4.7).

Input To estimate the height of a potato tuber, the system uses multiple images taken at an angle between 40 and 90 degrees to the board (Fig. 3.5.). The size of the

input images was reduced to 1280*720 pixels to reduce the computational complexity while maintaining enough detail for the feature point detection step.

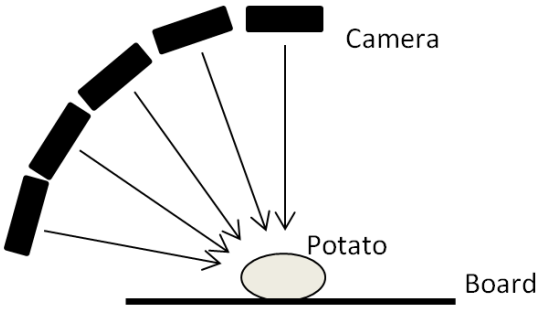


FIGURE 3.11: To estimate the height of the potato, multiple images are taken at angles ranging between 40 and 90 degrees to the board.

3.4.2 Pre-processing

The highly reliable fiducial square markers generated with the ArUco library (Garrido-Jurado et al., 2014) were used to find the outer contours of the board. Since we are only interested in recreating the board and the potato in 3D, pixels outside of the board contours were set to black to reduce the computational complexity.

3.4.3 Camera pose estimation

In Section 3.2.1 finding the intrinsic parameters of the camera was discussed. For multi-view stereo vision also finding the extrinsic parameters is needed. The extrinsic parameters describe the relative position of the camera and its view direction.

The extrinsic parameters can be expressed by the similarity transformation matrix T_{cm}, in which the position of the camera are the translation elements $t_{x,y,z}$ and the view direction is the rotation part r_{11-33}.

$$T_{cm} = \left[\begin{array}{c|c} R & T \\ \hline 0 & 1 \end{array}\right] = \left[\begin{array}{ccc|c} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{3.11}$$

The estimation of the camera pose was based on the highly reliable fiducial square markers generated with the ArUco library (Garrido-Jurado et al., 2014) of which the size is known. Based on these markers, the transformation matrices from the markers coordinates to the camera coordinates (T_{cm}) can be calculated (Equation 3.12). X_c

,Y_c and Z_c describe the coordinates of the camera and X_m, Y_m and Z_m describe the coordinates of the marker. Matrix R describes the rotation between the coordinate system of the marker and the coordinate system of the camera. The translation between the two coordinate sets is described by vector T.

Other methods exist for estimating the camera position without using markers such as Structure From Motion (SFM). These methods rely upon using the correspondence between images to estimate the camera's relative position. An advantage of square marker based camera position tracking is that it does not rely on correspondence between images. Furthermore, the square markers could also be used to easily determine the region of interest.

$$\begin{bmatrix} X_c \\ Y_c \\ Zc \\ 1 \end{bmatrix} = \left[\begin{array}{ccc|c} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix} = \begin{bmatrix} & \mathbf{R} & & \mathbf{T} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix} = T_{cm} \begin{bmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{bmatrix} \tag{3.12}$$

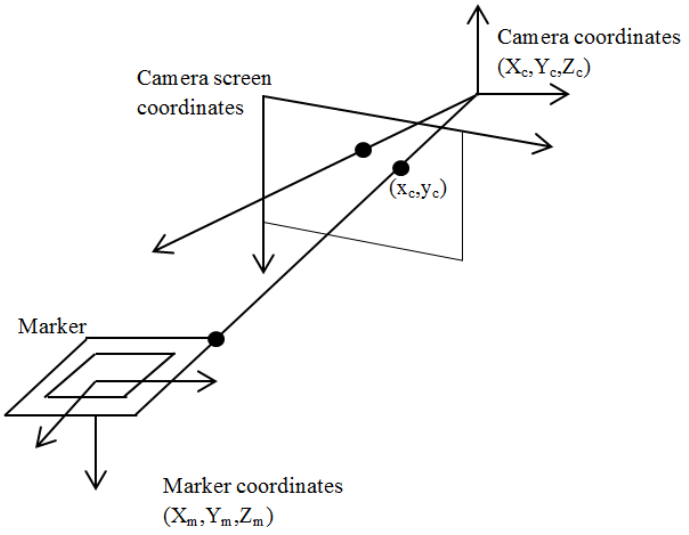


FIGURE 3.12: Relation between the marker coordinates and camera coordinates.

To determine the transformation matrix T_{cm} the a binary thresholded version of the input image is used to identify and locate each marker. Based on the found markers, the rotation matrix is calculated using the line segments of the marker, while the translation matrix is calculated based on the four corner points of the markers (the intersection points of the line segments).

Combining the intrinsic and extrinsic parameters gives the camera matrix P.

$$P = KT_{cm} \tag{3.13}$$

The camera matrix P is used to project a point from the real world $\mathbf{X}$ and project it into image coordinates $\mathbf{x}$ (both in homogenous coordinates).

$$\mathbf{x} = P\mathbf{X} \tag{3.14}$$

The result of the pose estimation can be seen in Figure 3.13.

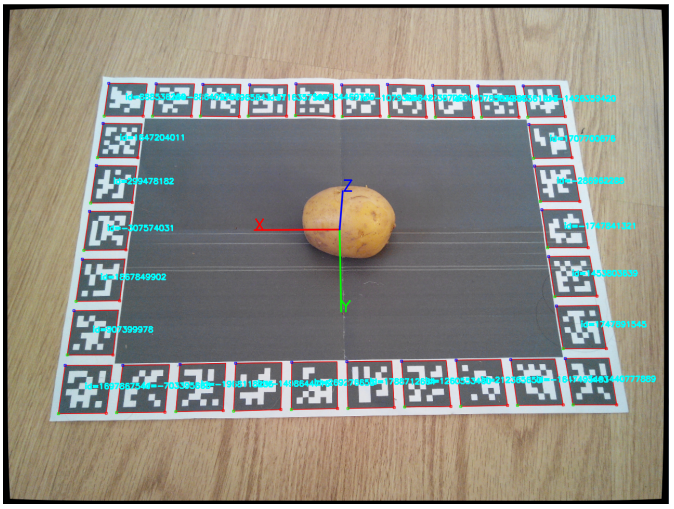


FIGURE 3.13: Camera pose estimation by using the maker board.

3.4.4 Feature point detection

The next step in the process of creating a 3D representation of a set of 2D images is finding and matching feature points between images. ORB (Rublee et al., 2011) was used as the feature detector and descriptor. ORB stands for Oriented FAST and Rotated BRIEF. As the name already suggests it builds on the FAST keypoint detector (Rosten and Drummond, 2006) and BRIEF descriptor (Calonder et al., 2010). ORB performance on par with the well known keypoint detectors and descriptors SIFT (Lowe, 1999)and SURF (Bay et al., 2006), while at the same time being more efficient (Rublee et al., 2011). Furthermore, it free to use in contrast to SIFT and SURF which are both restricted by licences.

For each image, a set of keypoints was determined by using FAST to find corners. Fast uses a series of comparisons between a pixel and a ring around the radius of the pixel to find corners. Harris corner measure was then used to find the top N points among the found keypoints.

FAST does not include any information about the rotation of the corner. To include rotation invariance, the ORB computes the intensity weighted centroid of the found keypoint patch with the located corner in the centre. The orientation is given by the direction of the vector of this corner point to the centroid. This calculation is depicted in the following equations:

$$m_{pq} = \sum_{x,y} x^p y^q l(x,y) \tag{3.15}$$

$$C = \left(\frac{m_1 0}{m_0 0}, \frac{m_0 1}{m_0 0} \right) \tag{3.16}$$

$$\theta = \arctan\left(\frac{m_{01}}{m_{10}}\right) \tag{3.17}$$

in which x and y represent the pixel location, m_{pq} represent the moment, $l(x,y)$ the intensity of a given point, C the centroid and finally θ the orientation.

Since BRIEF performs poorly with rotation, ORB uses the orientation of keypoints to 'steer' BRIEF. For any feature set of n binary tests at location (x_i, y_u) a $2xn$ matrix (S) is defined which contains the coordinates of these pixels. Based on the orientation of the patch (p) the rotation matrix is calculated which is used to rotate S to the steered (rotated) version S_p.

The BRIEF descriptor is then applied on S_p and records the binary string as ORB descriptor. Since ORB is a binary descriptor, the Hamming distance was used to match keypoints between images.

3.4.5 Triangulation

The next step taken to create a 3D representation based on the 2D images, was the triangulation of the found matching keypoints using the camera matrices. Triangulation is performed by Direct Linear Transform (DLT) Hartley and Sturm (1997).

From equation 3.14 it follows that:

$$\mathbf{x} \times P\mathbf{X} = 0 \text{ and } \mathbf{x'} \times P\mathbf{X'} = 0 \tag{3.18}$$

Therefore it is possible to obtain a new set of linear equations $A\mathbf{X} = 0$ where

$$A = \begin{bmatrix} x\mathbf{p}^{3T} - \mathbf{p}^{1T} \\ y\mathbf{p}^{3T} - \mathbf{p}^{1T} \\ x'\mathbf{p}^{3T} - \mathbf{p}'^{1T} \\ y'\mathbf{p}^{3T} - \mathbf{p}'^{1T} \end{bmatrix} \tag{3.19}$$

and $\mathbf{p}^{iT}$ is the i-th row of P. $\mathbf{X}$ can then be determined by finding the unit singular vector corresponding to the smallest singular value of A.

Using DLT the system created an initial point cloud by matching keypoints between pairs of images, and iteratively triangulate each matching pair of keypoints.

3.4.6 Bundle adjustment

The initial point cloud created in the triangulation step, might still contain a number of errors. A point triangulated using images form cameras 1 and 2 might not give the same 3D coordinate as the triangulation of the same point using images from camera 2 and 3 due to, for example, noise or errors in the measurement of the camera's position.

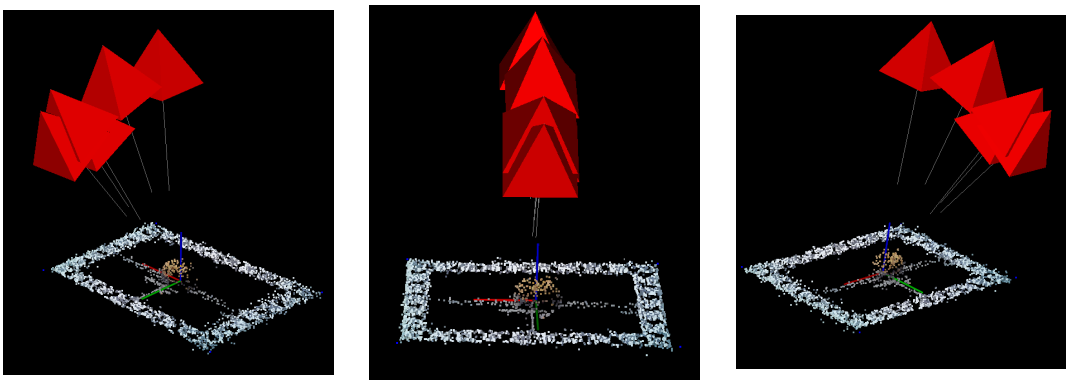
Bundle Adjustment (BA) was used to rectify theses errors. BA minimises the sum of the square distance between the i-th 3D points projection in image j ($\mathbf{x}_j^i$) and its reprojection $P_j\mathbf{X}_i$. In other by simultaneous adjusting the camera parameters BA tries to minimize:

$$\sum_{i=1}^{n}\sum_{j=1}^{m} v_{ij} d(Q(\mathbf{a}_j, \mathbf{b}_i), \mathbf{x}_j^i)^2 \tag{3.20}$$

in which n is the number of available points in m views, v_{ij} is a binary variable indicating whether or not point $\mathbf{X}_i$ is visible in camera, $Q(\mathbf{a}_j, \mathbf{b}_i)$ represents the reprojection, the vector $\mathbf{a}_j$ contains the camera parameters of camera j and the vector $\mathbf{b}_i$ contains the estimated non-homogenous 3D coordinates of the i-the point j (Triggs et al., 2000).

Minimizing formula 3.20 was performed with the Levenberg-Marquardt algorithm. This algorithm is a combination of the steepest decent and Gauss Newton methods for minimization (Marquardt, 1963).

The resulting point cloud was then visualized in the Point Cloud Library as can be seen in Figure 3.14.



(A) (B) (C)

FIGURE 3.14: Three views from a point cloud after BA. The red pyramids represent the camera positions.

3.4.7 Cluster extraction

The next step in the process is determining which points of the point cloud belong are part of a potato, and which points belong to the board. Since their might be more than one potato at a time, it is also important to determine which point belongs to which potato. This is achieved by clustering the total point cloud into separate point clouds for the board and point clouds for each potato.

First the point cloud data representing the board is segmented from the rest of the point cloud. Next, the points from the remaining cloud belonging to an individual potato are clustered.

Random sample consensus (RANSAC) (Fischler and Bolles, 1981) was used to find the points in the point cloud that belong to the board. RANSAC iteratively estimates the parameters of a mathematical model on the basis of a set of observed data which includes outliers.

For the point cloud resulting from the BA step, the points belonging to the board should be considered as inliers, while the point belonging to potatoes or noise are outliers.

The basic RANSAC algorithm is defined in Algorithm 1.

Algorithm 1 RANSAC

1: **repeat**
2: Select a random subset of *hypothetical inliers* from the original dataset
3: Fit a model to the set of hypothetical inliers
4: Test the rest of the dataset against this model. Points that fit the dataset well with a predefined tolerance ϵ are considered as part of the consensus set.
5: **until** The percentage of fitted points exceeds a predefined threshold θ or $n \geq N$
6: Re-estimate the model parameters using all the fitted inliers and terminate

One of the advantages of RANSAC is its ability to generate a robust estimation of the model parameters even when a significant number of outliers is present.

The Point Cloud Library was used to perform RANSAC to segment the board from the rest of the cloud. Since the board is a planar component, the build in model for planar segmentation was used.

To cluster the potatoes in the point cloud 3D grid subdivision based on Euclidean distances using an octree data structure was used (Rusu, 2010). The used algorithm is as follows:

Algorithm 2 Clustering

Create a Kd-tree representation for the input cloud dataset P.
Create an empty list of clusters C, and a queue Q of points which need checking.
for every point $p_i \in P$ **do**
    Add p_i to Q
    for every point $p_i \in Q$ **do**
        Search for the set P_i^k of neighbouring points of p_i in a sphere with radius $r < d_{th}$
        Create an empty list of clusters C, and a queue Q of points which need checking.
    end for
    if every point in Q has been processed **then**
        Add Q to the list of clusters C and empty Q.
    end if
end for

An implementation of this algorithm in PCL was used to cluster the potatoes within the point cloud.

Based on the clustered potatoes, the size of each potato was calculated. Of each potato point cloud cluster, the maximum z-value was determined. By relating the real-world size of the board to the point cloud representation of the board, a relation between the point cloud coordinates and real world coordinates could be defined. Using this relation, the height of each potato was calculated.

In Figure 3.15 an impression of the system in use is given.

3.5 Experimental setup

Within this section, the experimental setup is described which was used to evaluate the systems performance. Three separate experiments were conducted. The first experiment was aimed at testing the accuracy of the system while using a smartphone. The second part of the experiment was aimed at testing the accuracy while using the Google Glass.

FIGURE 3.15: Impression of the system in use.

The third experiment how well the system performs when estimating the size of multiple potatoes at the same time.

3.5.1 Image Acquisition

The experimental data for both experiments consisted of images of potatoes which were placed upon the marker board (Paragraph 3.1). Images were acquired using a Xiaomi RedMi smartphone equipped with a 8 megapixel camera, as well as a Google Glass equipped with a 5 megapixel camera.

For the first and second experiment, sequences of five picture were captured of each individual potato. These pictures were taken at increasing angles between 40 degrees and 90 degree to the to the centre of the board as depicted in Fig. 3.11. At least one picture of the sequence was taken perpendicular to the board in order to be able to assess the width of the potato in the 2D computer vision stage.

For the third experiment, multiple potatoes were placed on the board at the same time. Again pictures were taken at increasing angles between 40 degrees and 90 degree to the to the centre of the board as depicted in Fig. 3.11.

All images were taken during daytime in indirect sunlight without artificial lighting.

3.5.2 Data sets

Three data sets were collected. For the first dataset a sample of 10kg of potatoes was used. This sample contained 111 potatoes which resulted in a dataset consisting of 111

individual potatoes captured in sequences of 5 images taken with the Xiaomi RedMi smartphone. The second dataset consisted of 20 (2kg) individual potatoes captured in sequences of 5 images taken with the Google Glass, while the third dataset consisted of 54 potatoes (5kg) captured with the Xiaomi RedMi smartphone camera in 9 sequences of 5 images in which 6 potatoes were presented simultaneously the board. Within the last dataset, occlusion was present in a subset of the images due to the larger number of potatoes on the board.

Due to the quality sensors of the used cameras, motion blur could occur when the camera was slightly moved during the image capturing. Since these images can not be used to accurately estimate the 2D and 3D shape of the potato tuber, they were removed from the datasets.

3.5.3 Ground truth data

To be able to verify the accuracy of the system, ground truth data was collected. A calliper was used to accurately measure the length and width of each potato by hand. Furthermore, the square mesh size of each potato was measured with a square mesh size measuring tool (Fig. 1.1). All three measurements were performed with a precision of 1 millimetre.

3.5.4 Testing the system

The developed computer vision system was used to determine the width, height and square mesh size of each potato for all three datasets.

3.5.5 Evaluation

To evaluate the accuracy of the system for all three data sets, the measurements by the system were compared to the ground truth data. Both the mean absolute error in millimetres as the mean error as percentage were used to evaluate the performance of the computer vision system.

For the third dataset, the number of measured potatoes were compared to the actual number of potatoes present in the photographs to estimate the effect of the occlusion.

For each dataset, paired samples t-test were performed to determine whether the data measured with computer vision system significantly differed from the data measured with the calliper and square mesh size measuring tool.

Chapter 4

Results

4.1 Smartphone

In this section the results of the potato measurements based on images taken with the smartphone are discussed. In Appendix C the a complete overview of the research data can be found. Appendix B contains plots of the by hand measured data versus the computer vision data.

4.1.1 Height measurement

To compare the by calliper measured potato height with the potato height as measured by the computer vision system, a two-tailed paired-sample t-test was conducted. The by calliper measured potato heights ($M = 44.46, SD = 5.82$) did not significantly differ from the heights measured by the computer vision system ($M = 43.77, SD = 6.26$); $t(110) = 1.45, p = 0.15$.

The mean absolute error of the height measurement by the computer vision system was 3.91mm to the calliper measurements which is a mean percentage error of 8.80% (Table 4.1).

4.1.2 Width measurement

A two-tailed paired-samples t-test was conducted to compare by calliper measured potato widths to the potato widths as estimated by the computer vision system. The by calliper measured potato widths ($M = 52.50, SD = 7.33$) did significantly differ

from the widths measured by the computer vision system ($M = 53.78, SD = 9.57$); $t(110) = -2.97, p = 0.04$.

The computer vision system was able to measure the potato widths with a mean average error of 3.63mm to the calliper measurements and a percentage error of 6.86% (Table 4.1).

4.1.3 Square mesh size measurement

A two-tailed paired-samples t-test was also conducted to compare the square mesh sizes measured with the square mesh size tool were also compared to the measured square mesh sizes by the computer vision system. The square mesh sizes measured with the square mesh size tool ($M = 50.41, SD = 6.76$) did significantly differ from the square mesh size as measured by the computer vision system ($M = 49.16, SD = 7.20$); $t(110) = 3.36, p < 0.01$.

The computer vision system had a mean absolute error of 2.75mm when measuring the square mesh size of a potato compared to the calliper measurements, which is a percentage error of 5.55% (Table 4.1).

	Computer vision height	Computer vision width	Computer vision square mesh size
N	111	111	111
Mean absolute error (mm)	3.91	3.63	2.75
Mean percentage error (%)	8.80	6.86	5.55

TABLE 4.1: Deviation of the computer vision measurements based on smartphone images from the calliper measurements.

4.2 Google Glass

The results based on images taken with the Google Glass are discussed in this section.

4.2.1 Height measurement

A two-tailed paired-samples t-test was conducted to compare the by calliper measured potato heights to the potato heights as estimated by the computer vision system. There was no significant difference in the score for the calliper measured width ($M =$

$42.00, SD = 5, 75$) compared to the computer vision measured width ($M = 42, 18, SD = 6, 98$); $t(19) = -0.24, p = 0.81$.

The computer vision system had a mean absolute error of 3.33mm when measuring the square mesh size of a potato compared to the calliper measurements, which is a percentage error of 7.97%

4.2.2 Width measurement

Of the total dataset of 20 potatoes, the system was unable to measure the width of two potatoes. Analysis of the used images showed that the automated thresholding with the Otsu method did not result in a clear segmentation between the board and the potato. The lack of a clear segmentation was caused by slight changes in the lighting condition.

A two-tailed paired-samples t-test indicated that the by calliper measured potato widths ($M = 50.94, SD = 6, 32$) did significantly differ from the potato widths as measured by the computer vision system ($M = 52.87, SD = 7.26$); $t(17) = -3.48, p = 0.03$.

The measured mean absolute error of the height measurement by the computer vision system compared to the calliper measurements was 3.99mm, which is a percentage error of 7.82% (Table 4.2).

4.2.3 Square mesh size measurement

The calliper measured potato square mesh sizes were also compared to the measured square mesh sizes by the computer vision system by conducting a two-tailed paired-samples t-test. It showed that the potato square mesh sizes measured with the potato square mesh size tool ($M = 50.94, SD = 6.53$) did significantly differ from the measured square mesh sizes by the computer vision system ($M = 48.18, SD = 6.55$); $t(17) = 0.71, p = 0.49$.

For the square mesh size measurement, the mean absolute error of the computer vision measurements was 2.37mm compared to the calliper measurements. The mean percentage error was 4.75% (Table 4.2).

	Computer vision height	Computer vision width	Computer vision square mesh size
N	20	18	18
Mean absolute error (mm)	3.33	3.99	2.37
Mean percentage error (%)	7.97	7.82	4.75

TABLE 4.2: Deviation of the computer vision measurements based on Google Glass images from the calliper measurements.

4.3 Multiple potatoes

In this section an overview is given of the results from measuring multiple potatoes at the same time by the computer vision system.

4.3.1 Height measurement

Of the total number of 54 potatoes, the system was able to estimate the height for 52 potatoes (96.29%). Analysis of the used images showed that due to occlusion, the system was unable to create a 3D model of the missing potatoes due to a lack of keypoints.

A two tailed paired samples t-test was performed to compare the by calliper measured height and the height measured by the computer vision system. The by calliper measured height ($M = 45.12, SD = 4.97$) did significantly differ from the height determined by the computer vision system ($M = 41.65, SD = 6.44$); $t(51) = 5.13, p < 0.01$.

The mean absolute error for the height as determined by the computer vision system compared to the calliper height measurements was 4.81mm (10.82%) (Table 4.3).

4.3.2 Width measurement

The system was able to estimate the width of 53 of the total of 54 potatoes (98.18%). Analyses of the images showed that the system was unable to determine the width of one potato due to slight changes in lighting condition preventing a correct threshold determination and therefore prevented a correct segmentation of the potato from the background.

A two tailed paired samples t-test showed that the by calliper measured width ($M = 54.02, SD = 4.97$) did not significantly differ from the width determined by the computer vision system ($M = 53.27, SD = 9.27$); $t(52) = 0.67, p = 0.51$.

The mean absolute error for the width as determined by the computer vision system compared to the calliper width measurements was 5.63mm (10.45%) (Table 4.3).

4.3.3 Square mesh size measurement

Due to missing the height of 2 potatoes and the width of 1 potato, the system was able to estimate a square mesh size for 51 of the total of 54 potatoes (94.4%).

A two tailed paired samples t-test showed that the square mesh size measured with the tool ($M = 51.54, SD = 5.89$) did significantly differ from the square mesh size as determined by the system ($M = 48.02, SD = 7.32$); $t(50) = 5.02, p < 0.01$.

The mean absolute error for the width as determined by the computer vision system compared to the calliper width measurements was 4.44mm (8.60%) (Table 4.3).

	Computer vision height	Computer vision width	Computer vision square mesh size
N	53	54	52
Mean absolute error (mm)	4.81	5.63	4.44
Mean percentage error (%)	10.82	10.45	8.60

TABLE 4.3: Deviation of the computer vision measurements images from the calliper measurements during the simultaneous measurement of 6 potatoes.

Chapter 5

Discussion

Within this thesis we presented a system to measure potato tubers heights, widths and square mesh sizes based on images taken with a smartphone and Google Glass. Within this section the research question and sub-questions will be answered.

5.1 Height measurement

The first research sub-question was:

- "How accurate can the system measure the length of the minor axis (height) of a potato tuber?"

The results show that the computer vision system had an mean absolute error of 3.91mm (8.80%) for the Google Glass and 3.33mm (7.97%) for the smartphone.

For both the images taken with the Google Glass and the images taken with the smartphone, the measurements of the length of the minor axis by our computer vision system does not significantly differ from the hand measurements with the calliper. For parts, the mean average errors of 3.91mm (8.80%) and 3.33mm (7.97%) can be explained by measuring errors in the calliper measurements. Due to the irregular shape of potatoes, the calliper might not have been perfectly placed at the highest point of the potato.

Furthermore, the quality of the camera and the sharpness of the pictures influenced the photos. The Google Glass seemed to take slightly sharper pictures, compared to the Xiaomi RedMi which can explain the smaller average error of 3.33mm versus 3.91mm for the Xiaomi RedMi. Slight motion blur can cause distortion of the photo which influences the triangulation of the keypoints, and therefore the height measurement.

5.2 Width measurement

The second research sub-question was:

- "How accurate can the system measure the length of the intermediate axis (width) of a potato tuber?"

The computer vision system was able to measure the width of the potatoes with a mean absolute error 3.99mm (7.82%) for the Google Glass and 3.63mm (6.86%) for the smartphone datasets.

For both the width measurements based on the Google Glass images, as well as the smartphone images, the width measurements by the computer vision system differed significantly from those of the calliper measurements. The difference in measurements and average error for the Google Glass and smartphone were in parts caused by slight measurement errors in the calliper measurements.

Furthermore, the width measurements relied on images taken perpendicular to the surface of the board. Since the pictures were taken with the smartphone in the hand slight deviations from the desired angle occurred. Deviations were even bigger for the Google Glass since it is head-mounted which makes it more difficult to take perfect perpendicular images. This can explain the higher absolute mean error of the Google Glass compared to the Xiaomi RedMi.

Slight changes in lighting conditions hindered a perfect segmentation at times. Due to these changes, the outline of the potato was sometimes less clear in the saturation image, which caused slight deviations in the width estimation.

5.3 Square mesh size measurement

Our third sub-question was:

- "How accurate can the system derive the mesh size of a potato based on the measured length of the intermediate and minor axis of the potato tuber?"

The computer vision system was able to derive the square mesh size with an absolute mean errors of 2.75mm (5.55%) for the Google Glass and 2.37mm (4.75%)

The computer vision square mesh size measurements did not significantly differ from the hand measurements with the square mesh size tool for the Google Glass images, while it

did significantly differ for the smartphone images. This difference can be explained by the lower number of samples in the Google Glass dataset compared to the smartphone dataset.

Furthermore, in order to approximate the square mesh size the formula created by De Koning et al. (1994) was used (Section 2.2). This formula also introduced a small error. Even when using the widths and heights as measured with the calliper as input for the formula, the resulting square mesh sizes have an absolute mean error of 1,82mm (3.74%).

The mean absolute errors of 2.75mm (5.55%) for the Google Glass and 2.37mm (4.75%) for the smartphone show that on average the error becomes smaller in comparison to the individual width and height measurements due to the combination the width and height measurements for the square mesh in the formula by De Koning et al. (1994). Both mean absolute errors are within the range of the 3mm mean average error which was deemed acceptable in private discussions with the potato industry.

5.4 Multiple potato measurement

The fourth research sub-question was:

- "How well does the system handle measuring multiple potatoes at the same time?"

The results show that the system was able to handle multiple potatoes quite well. In 96.29% of the cases the system was able to estimate a potatoes height, in 98.18% of the cases the system was able to estimate a potatoes width, and in 94.4% of the cases a square mesh size could be determined. That the height of two potatoes could not be measured was caused by occlusion. Due to the occlusion, parts of the potato were not visible in some of the pictures, which reduced the number of keypoints that could be matched, which prevented the potato from being segmented from the board.

In contrast to the results of the Xiaomi RedMi when measuring one potato at a time, the estimated height of the potato, when measuring multiple potatoes at the same time by the computer vision system, did significantly differ from the measurement taken width the calliper.

Occlusion reduced the number of keypoints available per potato for the triangulation and bundle adjustment, which resulted in a less complete 3D model. Since the height measurement relies on the 3D model, an incomplete 3D model can cause a deviation in the height estimation.

There was no significant difference in the widths measured by the computer vision system compared to the widths measured by the calliper when measuring multiple potatoes at the same time. As with the width measurements of the single potatoes, the mean absolute error of 5.63mm (10.45%) was partly caused by the used images not being taken perfect perpendicular to the board. Furthermore, slight deviations in the detection of the outline of the potato on the board can further explain the mean absolute error.

The square mesh size as measured by the computer vision system did significantly differ from the square mesh size as determined by hand with the square mesh size tool. The mean absolute error of 4.44mm (8.60%) lies outside the range of 3mm as deemed acceptable in private discussions with the potato industry. Like the square mesh size measured with the Google Glass, the mean absolute error can be explained by deviations in the measurement of the potatoes widths and heights, as well as deviations resulting from the used formula (De Koning et al., 1994) for the square mesh size estimation.

5.5 Research Question

The main research question was:

- "What is the viability of a computer vision potato grading system for mobile devices?"

The answers on the sub-question show that the computer vision potato grading system was able to determine the square mesh size of a single potato within the acceptable range of 3mm for the mean absolute error as determined in private conversations with representatives from the potato industry. Although the accuracy declines when more than one potato is measured at a time, the results show that mobile computer vision could be a viable option for automated potato grading.

This thesis shows that a dedicated computer vision setup does not have to be necessary for potato size grading. By using images captured with a relatively cheap smartphone or smartglass, the system was able to size grade potatoes at a much lower cost than a dedicated computer vision setup. Although smartglasses such as the Google Glass are at the time of writing more expensive than most smartphones, they make it possible to capture the images while keeping your hands free. Using the smartphone has as advantage of grading potatoes at an even lower cost.

We can conclude that the system presented in this thesis is a viable way to grade potatoes based on their square mesh size. Future work might further improve the accuracy and usability of the system. The future of agricultural grading can be mobile!

5.6 Improvements and future research

One of the main reasons for the estimation error of the square mesh size seems to be the quality of the camera. Lower quality cameras suffer from more blur and noise which influence the size estimation. In recent years, the quality of smartphone/smartglasses cameras have been rapidly increasing, while the costs dropped. For future research, using cameras that produce sharper images in might further improve the accuracy of the system. When the images are sharper, the keypoint detection will be more precise, resulting in a better triangulation.

Furthermore, sharper images can also improve the segmentation of the potato from the board in the 2D computer vision phase. The outline of the potato on the board will be clearer on sharper images, which makes the segmentation, and therefore the width estimation, more precise.

The width measurement accuracy can also be improved by using segmentation methods which are less influenced by the lighting conditions. K-means segmentation or watershed can further improve the quality of segmentation since they are less prone to error due to changes in lighting conditions, although the computation load will increase.

This thesis has shown the possibility of estimating potato sizes based on 2D and 3D computer vision in a mobile application. Since the 3D computer vision step relies on non-potato specific keypoints, the system should be easily adaptable for grading other kind of agricultural products. Future research is necessary to investigate whether this is indeed possible.

Bibliography

E. Abdel-Hakim and A. Farag. CSIFT: A SIFT Descriptor with Color Invariant Characteristics. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2 (CVPR'06)*, 2:1978–1983, 2006. doi: 10.1109/CVPR.2006.95.

A. Al-Mallahi, T. Kataoka, H. Okamoto, and Y. Shibata. Detection of potato tubers using an ultraviolet imaging-based machine vision system. *Biosystems Engineering*, 105(2):257–265, Feb. 2010. ISSN 15375110. doi: 10.1016/j.biosystemseng.2009.11.004.

H. Bay, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. *Computer Vision–ECCV 2006*, 3951:404–41, 2006.

S. Beucher and C. Lantuejoul. Use of watersheds in contour detection. In *Int. Workshop Image Processing*, pages 2.1 –2.12. Real-Time Edge and Motion Detection/Estimation, 1979.

M. A. Boden. *Mind as machine: A history of cognitive science*, volume 1. Oxford University Press, 2006.

G. Bradski. Opencv. *Dr. Dobb's Journal of Software Tools*, 2000.

M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: Binary robust independent elementary features. In *Computer Vision–ECCV 2010*, pages 778–792. Springer, 2010.

J. Canny. A computational approach to edge detection. *IEEE transactions on pattern analysis and machine intelligence*, 8(6):679–98, June 1986. ISSN 0162-8828.

K.-T. Cheng and Y.-C. Wang. Using mobile gpu for general-purpose computing–a case study of face recognition on smartphones. In *VLSI Design, Automation and Test (VLSI-DAT), 2011 International Symposium on*, pages 1–4. IEEE, 2011.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sept. 1995. ISSN 0885-6125. doi: 10.1007/BF00994018.

C. De Koning, L. Speelman, and H. De Vries. Size grading of potatoes: Development of a new characteristic parameter. *Journal of agricultural engineering research*, 57(2): 119–128, 1994.

M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

Y. Freund and R. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. *Computational learning theory*, pages 1–34, 1995.

S. Garrido-Jurado, R. M. noz Salinas, F. Madrid-Cuevas, and M. Marí-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280 – 2292, 2014. ISSN 0031-3203.

J. Gauch. Investigations of image contrast space defined by variations on histogram equalization. *CVGIP: Graphical Models and Image Processing*, 54(4):269–280, 1992.

C. Harris and M. Stephens. A Combined Corner and Edge Detector. *Procedings of the Alvey Vision Conference 1988*, pages 23.1–23.6, 1988. doi: 10.5244/C.2.23.

R. I. Hartley and P. Sturm. Triangulation. *Computer vision and image understanding*, 68(2):146–157, 1997.

R. Hasankhani and H. Navid. Potato Sorting Based on Size and Color in Machine Vision System. *Journal of Agricultural Science*, 4(5):235–244, Mar. 2012. ISSN 1916-9760. doi: 10.5539/jas.v4n5p235.

P. H. Heinemann, N. P. Pathare, and C. T. Morrow. An automated inspection station for machine-vision grading of potatoes. *Machine Vision and Applications*, 9(1):14–19, 1996.

G. Jahns, H. Møller Nielsen, and W. Paul. Measuring image analysis attributes and modelling fuzzy consumer aspects for tomato quality grading. *Computers and Electronics in Agriculture*, 31(1):17–29, Mar. 2001. ISSN 01681699. doi: 10.1016/S0168-1699(00)00171-X.

J. Jin, J. Li, G. Liao, X. Yu, and L. C. C. Viray. Methodology for potatoes defects detection with computer vision. In *International Symposium on Information Processing, Huangshan*, pages 346–351, 2009.

L. Juan and O. Gwun. A comparison of sift, pca-sift and surf. *International Journal of Image Processing (IJIP)*, 3(4):143–152, 2009.

M. Juneja and P. S. Sandhu. Performance evaluation of edge detection techniques for images in spatial domain. *methodology*, 1(5):614–621, 2009.

Y. Ke and R. Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–506. IEEE, 2004.

R. Kemp, N. Palmer, T. Kielmann, and H. Bal. Cuckoo: a computation offloading framework for smartphones. In *Mobile Computing, Applications, and Services*, pages 59–79. Springer, 2012.

M. Khojastehnazhand, M. Omid, A. Tabatabaeefar, et al. Determination of orange volume and surface area using image processing technique. *International Agrophysics*, 23(3):237–242, 2009.

A. B. Koc. Determination of watermelon volume using ellipsoid approximation and image processing. *Postharvest Biology and Technology*, 45(3):366–371, Sept. 2007. ISSN 09255214. doi: 10.1016/j.postharvbio.2007.03.010.

V. Leemans, H. Magein, and M.-F. Destain. Defects segmentation on 'Golden Delicious' apples by using colour machine vision. *Computers and Electronics in Agriculture*, 20 (2):117–130, July 1998. ISSN 01681699. doi: 10.1016/S0168-1699(98)00012-X.

Q. Li, M. Wang, and W. Gu. Computer vision based system for apple surface defect detection. *Computers and Electronics in Agriculture*, 36(2-3):215–223, Nov. 2002. ISSN 01681699. doi: 10.1016/S0168-1699(02)00093-5.

X. Liming and Z. Yanchao. Automated strawberry grading system based on image processing. *Computers and Electronics in Agriculture*, 71:S32–S39, Apr. 2010. ISSN 01681699. doi: 10.1016/j.compag.2009.09.013.

D. Lowe. Object recognition from local scale-invariant features. *Proceedings of the Seventh IEEE International Conference on Computer Vision*, pages 1150–1157 vol.2, 1999. doi: 10.1109/ICCV.1999.790410.

D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):pp. 431–441, 1963. ISSN 03684245.

J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761–767, Sept. 2004. ISSN 02628856. doi: 10.1016/j.imavis.2004.02.006.

K. Mikolajczyk and C. Schmid. Performance evaluation of local descriptors. *IEEE transactions on pattern analysis and machine intelligence*, 27(10):1615–30, Oct. 2005. ISSN 0162-8828. doi: 10.1109/TPAMI.2005.188.

H. P. Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. Technical report, DTIC Document, 1980.

J.-M. Morel and G. Yu. ASIFT: A New Framework for Fully Affine Invariant Image Comparison. *SIAM Journal on Imaging Sciences*, 2(2):438–469, Jan. 2009. ISSN 1936-4954. doi: 10.1137/080732730.

E. N. Mortensen, H. Deng, and L. Shapiro. A sift descriptor with global context. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 184–190. IEEE, 2005.

V. Narendra and K. Hareesha. Quality inspection and grading of agricultural and food products by computer vision-a review. *International Journal of Computer Applications*, 2(1):43–65, 2010.

N. Otsu. A threshold selection method from gray-level histograms. *Automatica*, 20(1): 62–66, 1975.

J. Paliwal, N. Visen, D. Jayas, and N. White. Cereal Grain and Dockage Identification using Machine Vision. *Biosystems Engineering*, 85(1):51–57, May 2003. ISSN 15375110. doi: 10.1016/S1537-5110(03)00034-5.

I. Paulus and E. Schrevens. Shape Characterization of New Apple Cultivars by Fourier Expansion of Digitized Images. *Journal of Agricultural Engineering Research*, 72(2): 113–118, Feb. 1999. ISSN 00218634. doi: 10.1006/jaer.1998.0352.

I. Paulus, R. De Busscher, and E. Schrevens. Use of Image Analysis to Investigate Human Quality Classification of Apples. *Journal of Agricultural Engineering Research*, 68(4): 341–353, Dec. 1997. ISSN 00218634. doi: 10.1006/jaer.1997.0210.

J. Prewitt. Object enhancement and extraction. *Picture processing and Psychopictorics*, 10(1):15–19, 1970.

T. Ridler and S. Calvard. Picture thresholding using an iterative selection method. *IEEE transactions on Systems, Man and Cybernetics*, 8(8):630–632, 1978.

R. Rios-Cabrera. ANN analysis in a vision approach for potato inspection. *Journal of applied ...*, 6(2):106–119, 2008.

R. Rios-Cabrera, I. Lopez-Juarez, and H. Sheng-Jen. Ann analysis in a vision approach for potato inspection. *Journal of applied research and technology*, 6(2):106–117, 2008.

M. Riquelme, P. Barreiro, M. Ruiz-Altisent, and C. Valero. Olive classification according to external damage using image analysis. *Journal of Food Engineering*, 87(3):371–379, Aug. 2008. ISSN 02608774. doi: 10.1016/j.jfoodeng.2007.12.018.

E. Rosten and T. Drummond. Machine learning for high-speed corner detection. *Computer Vision–ECCV 2006*, pages 1–14, 2006.

E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. *2011 International Conference on Computer Vision*, pages 2564–2571, Nov. 2011. doi: 10.1109/ICCV.2011.6126544.

R. B. Rusu. Semantic 3d object maps for everyday manipulation in human living environments. *KI-Künstliche Intelligenz*, 24(4):345–348, 2010.

R. B. Rusu and S. Cousins. 3d is here: Point cloud library (pcl). In *International Conference on Robotics and Automation*, Shanghai, China, 2011 2011.

R. E. Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

P. C. Struik, J. Haverkort, D. Vreugdenhil, C. B. Bus, and R. Dankert. Manipulation of tuber-size distribution of a potato crop. *Potato Research*, 33(4):417–432, Dec. 1990. ISSN 0014-3065. doi: 10.1007/BF02358019.

D.-W. Sun. Computer vision—an objective, rapid and non-contact quality evaluation tool for the food industry. *Journal of Food Engineering*, 61(1):1–2, Jan. 2004. ISSN 02608774. doi: 10.1016/S0260-8774(03)00182-1.

Y. Tao, C. Morrow, P. Heinemann, and H. Sommer. Fourier-based separation technique for shape grading of potatoes using machine vision. *Transactions of the ASAE*, 38(3): 949–957, 1995.

B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment — a modern synthesis. In *Vision Algorithms: Theory and Practice*, volume 1883 of *Lecture Notes in Computer Science*, pages 298–372. Springer Berlin Heidelberg, 2000. ISBN 978-3-540-67973-8. doi: 10.1007/3-540-44480-7_21.

L. Vincent and P. Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE transactions on pattern analysis and machine intelligence*, 13(6):583–598, 1991.

J. Wu, Z. Cui, V. S. Sheng, P. Zhao, D. Su, and S. Gong. A Comparative Study of SIFT and its Variants. *Measurement Science Review*, 13(3):122–131, Jan. 2013. ISSN 1335-8871. doi: 10.2478/msr-2013-0021.

Z. Xiao-bo, Z. Jie-wen, L. Yanxiao, and M. Holmes. In-line detection of apple defects using three color cameras system. *Computers and Electronics in Agriculture*, 70(1): 129–134, Jan. 2010. ISSN 01681699. doi: 10.1016/j.compag.2009.09.014.

Q. Yang. Apple Stem and Calyx Identification with Machine Vision. *Journal of Agricultural Engineering Research*, 63(3):229–236, Mar. 1996. ISSN 00218634. doi: 10.1006/jaer.1996.0024.

Z. Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334, 2000.

L. Zhou, V. Chalana, and Y. Kim. PC-based machine vision system for real-time computer-aided potato inspection. *International Journal of Imaging Systems and Technology*, 9(6):423–433, 1998. ISSN 0899-9457. doi: 10.1002/(SICI)1098-1098(1998)9:6⟨423::AID-IMA4⟩3.0.CO;2-C.

Appendix A

Review

A.1 General overview of computer vision

The field of computer vision has been expanding at a rapid rate in the last couple of decades. Digital cameras have become cheap and widespread. This has resulted in a large availability of image data which can be processed by computer vision for automatic detection, recognition and classification purposes. Combined with the ever increasing computer power at decreasing costs, this trend has made it possible to use computer vision for more and more applications.

Applications of computer vision include inspecting and measuring products, recognizing faces and license plates and classifying images based upon the scene they depict. While these application vary widely, they are often build upon the same principals and techniques. Therefore a lot of libraries, toolkits and frameworks have been built to facilitate the implementation of computer vision techniques. In this appendix an overview of different basic computer vision techniques is described. Furthermore, an overview is given of a wide variation of well-known computer vision libraries, toolboxes and frameworks. At the end of this appendix a decision matrix is given which is based upon the functionality of each of the libraries, toolboxes and frameworks. This decision matrix is built to facilitate the decision upon which software package could be used for specific computer vision purposes.

A.1.1 Basic structure of computer vision applications

Most computer vision applications use the same basic structure. During the first step, images or video has to be acquired. This can be done with the help of all kinds of video and image taking equipment. While the process of capturing image data might seem

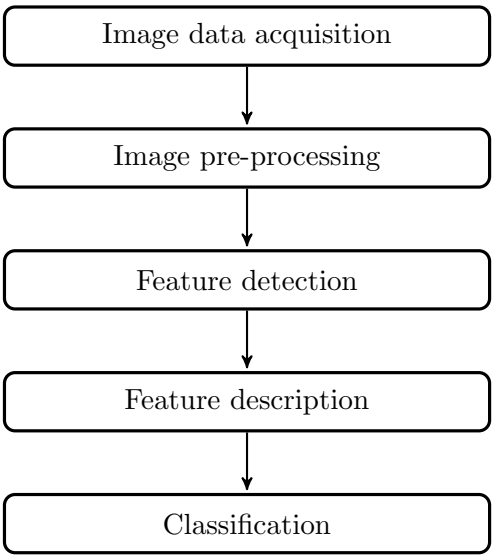


FIGURE A.1: Processing steps on which most computer vision applications are based.

straight forward, it can suffer from the large variability within image data. Changes in lighting conditions, changes in the angle of the camera, noise and occlusion by other objects can make the task of computer vision a lot harder. If possible, it is therefore necessary to create a set-up in which the acquisition of the image data facilitates the future computer vision processes by keeping the image data as constant as possible. This might be challenging however in some cases. Especially if the camera for image acquisition is mobile.

The next step in often taken by computer vision applications is eliminating the variability as best as possible with the help of pre-processing. The effect of noise is cancelled out as best as possible, as well as the effect of differences in the image data caused by changing lighting conditions.

During the third step in most computer vision applications, key regions or key points in an image are detected based upon features such as edges, corners and connected regions.

The keypoints detected in the third step are described in the fourth step with feature descriptors in order to be able to compare the keypoints to other images or known models of objects.

Based on the keypoint detection and descriptor steps, a classification can then be made of what is actually depicted in (parts of) the image data.

The steps most often taken in computer vision are depicted in Figure A.1.

A.2 Image data acquisition

The first step within computer vision applications is most of the times the acquisition of image or video data, although in some cases the image/video data might be pre-existing. Depending on the source, this data can be either in the form of still images, or in the form of video. If the data is not pre-existing, the choice for which type of image data (video/still images) depends on the specific computer vision application. In essence though, video data is build up from still images, which means that a lot of the same computer vision techniques can be used for both still images, as well as video (in the form of a sequence of still images). Therefore, in this section the focus will lie upon the usage of computer vision techniques for still images.

When capturing image data, one has to take into account the variabilities that come into play. The device that is used to capture image data can be an important factor in the rest of the computer vision process. Depending on the application the quality demands for the capturing device can vary. Capturing images with high detail and a low noise ratio can aid the rest of the computer vision process since less pre-processing steps are necessary. Furthermore, variability can be introduced by changes in the environment such as the illumination conditions. Keeping this variability as low as possible is often preferable to aid the computer vision process since less pre-processing steps are necessary.

A.3 Pre-processing

As described in the introduction, the pre-processing step is often necessary to remove variability (in the form of noise or changes in lighting) from an image. Several different algorithms have been developed to this extend.

A.3.1 Noise Removal

Digitally captured images contain almost always some noise. These random fluctuation in image colour and brightness can be a product of the sensor or circuitry of the capturing device due to electronic noise. The more noise an image gets, the harder it can become to use the image in computer vision applications such as object recognition. Therefore, several techniques have been developed that try to remove or reduce the amount of noise. One of the most basic ways to remove noise is by averaging the image.

Linear Filter Applying a linear filter is a very simple way of removing noise by averaging an image. The linear filter averages the intensity value of a pixel by using

the intensity value of its neighbours. By increasing the the filter size a less noise and smoother image can be created, though at the same time details will be lost. By adjusting the filters weighting value, the filter can be used to remove different types of noise. Increasing the weighting value of the central pixel will make the central pixel dominate the averaging.

Median Filter The median filter uses the median intensity value of pixels in the filter size to determine the intensity value for the central pixel. The median filter will leave the brightness differences in the picture intact and will, in contrast to the linear filter, not shift the edges of the image.

Gaussian Smoothing Gaussian smoothing removes noise within an image by using a Gaussian function. The effect of Gaussian smoothing is a less detailed image in which larger structures within the image can become more enhanced. The effect can be visualized as looking through a translucent window at the image.

A.3.2 Enhancing Contrast

When an image has low contrast, there are several techniques to boost the contrast.

Histogram Scaling Histogram Scaling is used most often used to increase the contrast of an image by increasing the difference between two neighbouring intensity values. Although it can also be used the other way round. The transform function that is used for the histogram scaling can be linear, as well as non-linear, and can be one-to-one or multiple-to-one.

Histogram Equalization Most transform functions for histogram scaling are limited to proposed cases. Histogram equalization has been developed to create a flexible and optimal function that can be employed for different types of images. Histogram equalization generates a much more uniform histogram by spreading out the number of pixels at the peaks of the original histogram and selectively compressing the number of pixels at the valleys (Gauch, 1992).

A.4 Feature Detection

Processing visual data can be quite computer intensive due to the large input size. Reducing the size of the input data can therefore be useful. Feature detection techniques can be used to reduce the input size by making use of the fact that often only parts of the image are of interest. Feature detection tries to find those parts of the visual input data that contain information that is of interest. This prevents wasting computational resources on data which is not of interest.

A.4.1 Edge Detection

One important feature of most objects, are the edges of the object. They can define the difference between the object and its background. Furthermore, they can be used to distinguish characteristic patterns, such as the stripes of a zebra, of an object. In an image, these edges are defined as those places in which the image brightness has discontinuities.

Multiple different methods to find edges have been developed. One big advantage of using edge detection, is that the redundancy of an image is reduced by a large vector. This reduction is useful since it reduces both the storage space of the image, as well as the computational complexity for future operations on the image.

Edge detection operators A classical method for detecting edges is by convolving an image with a edge detection operator (a 2D-filter). These operators are constructed in such a way that they are sensitive to large gradients in the image, while they return zero values for the uniform regions of the image. More formally, an edge operator is a neighbourhood operation which determines to which extend a simple arc passing through the pixel can partition each pixel's neighbourhood in such a way that pixels on one side of the arc will have one predominant value, while the pixels in the neighbourhood on the other side will have another classes predominant value.

Several different types of edge detection operators exist such as gradient operators and Laplace Operator. Gradient operators use the first or second derivative of the grey level of an image as basis. The first derivative will mark edge points of varying strength depending on the steepness of the grey level changes. The second derivative will return an impulse on either side of the edge. A line can then be drawn between the two impulses. The point at which the line crosses the zero axis is then at the centre of the edge. This makes it possible to detect edges on a sub-pixel scale.

Classical Edge detection Most traditional edge detection operators, such as the Prewitt (Prewitt, 1970) and Sobel operator, use the first derivative is used to detect edges. If the gradient is above a certain threshold, there is an object in the image. The Prewitt and Sobel operator both use two 3x3 kernels, one for the x axis and one for the y axis, to approximate the first order derivative. This approximation is done by convolving the image with these two kernels. These operators are relatively simple and are able to also detect the orientation of the edges, but are sensitive to noise which introduces inaccuracy.

Laplacian Edge Detection Laplacian operators use the second derivative of the grey level of an image to determine the edges. In contrast to the Prewitt and Sobel operator it only uses one kernel which is convolved with the image. This makes it faster to run compared to the Prewitt and Sobel operator. By testing a wider area around each pixel, Laplacian operators are able to find edges and there orientation with more precision but suffer but are very sensitive to noise.

Laplacian of Gaussian In order to tackle the problem of noise for the Laplacian operator, the Laplacian of Gaussian (LOG) was developed. This operator first uses a Gaussian filter to smooth the image, after which the Laplacian operator can be used to detect the edges. An example of an edge detection algorithm that makes use of LOG is the Marr-Hildreth algorithm (Prewitt, 1970).

Canny Edge Detection One of the most well used edge detectors was developed by Canny (1986). This detector uses a couple of step to determine the edges in an image. Since edge detection can be heavily influenced by image noise, the first step of the canny edge detector is to remove noise with the help of a Gaussian filter. Next, the Canny edge detector tries to find the intensity gradient of the image using four filters to detect horizontal, vertical and diagonal edges since an image can contain edges in multiple directions. In order to thin the edges, non-maximum suppression is used.

The larger the intensity gradients within the image, the larger the change that that image gradient is an edge. Determining the threshold at which a given intensity grading switches from corresponding to an edge or not can be hard. Canny Edge detection uses thresholding with hysteresis in order to circumvent this problem. This technique requires both a low and a high threshold. It is based on the assumption that important edges are along continuous curves in the images and that therefore faint lines can be followed and that pixels that do not constitute a line but do produce a high gradient difference can be discarded.

A.4.1.1 Advantage en disadvantages of different edge detection methods

Table A.1 gives an overview of the pros and cons of different well used edge detection operators. As can be seen in this table each edge detection technique has its own pros and cons. Based on the demands of the specific computer vision application a choice for a technique can be made.

Operator	Advantages	Disadvantages
Classical (Prewitt, Sobel)	Relatively simple,detects edges and their orientations	Inaccurate and sensitive to noise
Laplacian	Detects edges and their orientations.	Inaccurate and sensitive to noise
LOG (Marr-Hildreth)	Finds correct location of edges (to a sub-pixel scale). Tests a wider area around the pixel.	Tends to malfunction at corners, curves and where the grey level intensity function varies. Due to the use of the Laplacian filter, it is unable to find the orientation of edges
Gaussian (Canny, Shen-Castan)	Makes use of probability to find the error rate, localization and response. Improves signal to noise ratio for better detection in noise conditions.	Due to its complexity computationally intensive. Suffers from false zero crossings

TABLE A.1: An overview of the pros and cons of different edge detection operators.

An overview of the result of the different edge detection algorithms can be seen in Figure A.2.

A.4.1.2 Thresholding

Thresholding divides the histogram in one (bi-level thresholding) or multiple (multilevel thresholding) values depending on the characteristics of the histogram. In bi-level thresholding, pixels with intensity values below the threshold are considered as background, while pixels with intensity values above the threshold are considered as objects. In

FIGURE A.2: A comparison of the different edge detection techniques, Sobel (b), Prewitt (c), Laplacian (d), Laplacian of Gaussian (e) on an image (a) (Juneja and Sandhu, 2009).

multilevel thresholding, different classes of objects can be defined for intensity values between two thresholds. In theory, the number of thresholds can be increased indefinitely, though due to the exponential increase in computer load with each additional threshold, usually only bi- and tri-level thresholding is used.

The value used as threshold can be either a fixed value (global thresholding), or a value that depends on the local characteristics of the pixels (adaptive thresholding). Since the local threshold does not only depend on the histogram of the image but also the size of the image, it is more computational intensive compared to fixed value thresholding. Since adaptive thresholding takes into consideration the local characteristics of the pixels, it is better able to cope with changes in illumination throughout the image.

Threshold selection The used global threshold for segmentation can either be chosen manually or automatic. Manual threshold selection is the most basic and simple method. It depends on the researcher/user to select the threshold with the help of a graphical user interface. The dependency of this method on the researcher/user is also the downside of this method. It needs human supervision.

Several different techniques have been developed to automatically determine the threshold. One of the earliest methods is the isodata algorithm which was originally proposed by Ridler and Calvard (1978). In the isodata algorithm, the initial threshold is first guessed (often on base of the average intensity value of the image) and then used to segment the image into two classes(class A and class B). The algorithm then computes the average intensity value for both classes (ma and mb). The average of ma and mb is used to update the threshold. This update process is repeated until convergence is achieved.

Automatically determining the threshold can also be done with the help of clustering. The main clustering method for threshold selection is k-means clustering. K-means clustering tries to select threshold(s) which divides the image in k classes. Pixels are assigned to each class in such a way that the intensity of the pixel is closer to the average intensity of the assigned class then to all the other classes.

Region Based Segmentation Growing-and-merging (GM) and splitting-and merging (SM) are two region based segmentation techniques that are well known. GM works by selecting an initial pixel as start point of a growing region. Neighbouring pixels with similar characteristics (such as intensity and texture) are iteratively merged with growing region until no more pixels can be merged. In the next step, a pixel which has not been merged with a region, is selected as the start point of a new growing region. This procedure is repeated until there are no more pixels left that do not belong to a region.

SM regards the whole image initially as one big region. Iteratively SM splits this region in smaller regions with uniform image characteristics such as colour, gradient and texture. The segmentation stops when there are no more regions that have non-uniform characteristics to split.

Watershed The technique of Watershed was first introduced by Beucher and Lantuejoul (1979). The idea behind the technique of Watershed is to see an image as a topographical surface in which the elevation of each point depends upon the intensity value of the corresponding pixel. The watershed algorithm developed by Vincent and Soille (1991) works by flowing 'water' over the topographical surface. The local mini will become basins. When water from two basins meets, a damn is build. This way regions are created, and the image is simultaneously segmented.

A.4.2 Points of Interest detection

When humans look at a certain scene, they automatically focus upon certain parts of this scene. For example if there is a person in a scene, we normally tend to focus more upon the face of that person instead of focussing on his hands. Within this face we often focus more upon the eyes instead of the chin. Within computer vision, these points are called points of interest. Corners and blobs are such points of interest.

A.4.2.1 Corner

A corner in an image can be defined as the point at which there is a strong two-dimensional intensity change which makes the point well distinguishable from its neighbouring points. Corners are widely used as points of interest since they correspond with image locations with a high information content which can be matched between images. Moravec Corner Detector

Moravec (1980) developed one of the first corner detectors. Its method considers a local window in the image which is shifted by small amount in four different directions. The sum of square differences is then used to calculate the error between shifted patches with the original image. A flat patch will produce a low value for all the shift, straight edge will only yield a large error in the direction perpendicular to the edge, while a corner should produce a large change in any direction since it has two non-parallel edges.

Harris & Stephens / Plessey operator Harris and Stephens (1988) build a corner detector which is based upon Moravec corner detector. Instead of directly computing the sum of squared differences, the Harris detector uses a first order approximation of patch shift to calculate the sum of squared differences.

A.4.2.2 Blobs

Within computer vision, blobs are regions of pixels which differ in properties, such as colour or brightness, from the surrounding area of pixels.

The Difference of Gaussians approach The difference of Gaussian approach for blob detection works by smoothing the image multiple times with a Gaussian filter at different scales. The filtered images at each scale will be subtracted from the image at the previous scale. Blobs can then be detected based on the local extrema.

A.5 Feature Descriptors

Feature descriptors are used to describe and extract the keypoints and regions of interest of an image in order to be able to compare those keypoints and regions to other keypoints and regions.

Scale invariant feature transform (SIFT) Scale invariant feature transform was developed by Lowe (1999). It is a feature descriptor which is able to robustly detect objects in an image, even if these objects are partly occluded or among clutter. The four major stages of SIFT are: scale-space extrema detection, keypoint localization, orientation assignment and keypoint descriptors. During the first stage, a Difference-of-Gaussian (DOG) function is used to identify potential points of interest which are invariant to both scale and orientation. DoG was chosen to be used instead of Gaussian due to the lower demand it poses on computational resources.

During the the keypoint localization step, the points that are low of contrast, and the ones which are edge responses, are eliminated. Next, the dominant orientation of each keypoint is determined based on its local image patch. Based on the orientation, scale and location of each keypoint SIFT contracts a canonical view for the keypoints which is invariant to similarity transform.

In the final stage, a local image descriptor is built for each keypoint based upon the image gradients in its local neighbourhood. The summation of gradients strengths of a keypoint is used by SIFT to vote in a histogram for every neighbourhood according to the gradient directions.

PCA-SIFT Ke and Sukthankar (2004) developed an adaptation of SIFT based on Principal Component Analysis (PCA) called PCA-SIFT. PCA is a widely used effective method for data dimensionality reduction. The first three stages of PCA-SIFT are the same as the first three stages of SIFT (scale-space extrema detection and keypoint localization). In the final stage (keypoint descriptors) PCA-SIFT uses PCA instead of the gradient histogram method of SIFT.

Gradient location-orientation histogram (GLOH) Gradient location-orientation histogram (GLOH) is an extension to the SIFT descriptor in which the size of the descriptor is reduced with Principal Component Analysis (Mikolajczyk and Schmid, 2005).

Speeded Up Robust Features (SURF) Inspired by SIFT, Bay et al. (2006) created a feature descriptor called Speeded Up Robust Features (SURF). SURF can be divided into two stages, namely, keypoint detection and keypoint description. At its first stage it uses integral images which give a fast approximation of the Laplacian of Gaussian using a box filter. In order to detect the keypoints, determinants of the Hessian matrix are used. This way, SURF builds its scale space by keeping the image size constant and

only varying the filter size. The result of this first stage makes SURF invariant to both scale as well as location.

During the second and final stage, each keypoint gets assigned a reproducible orientation with the help of Haar wavelets to make the keypoints orientation invariant. To extract the descriptors, the sum of the wavelets response is used.

Global-SIFT (GSIFT) Mortensen et al. (2005) proposed a variation on SIFT called GSIFT which takes into account the global context by adding a global texture vector to the basis of SIFT.

Coloured-SIFT (CSIFT) During keypoint detection, SIFT uses only grayscale information of an image. Abdel-Hakim and Farag (2006) developed CSIFT is an adaptation of SIFT which uses the colour invariance of the image during keypoint detection.

Affine-SIFT (A-SIFT) Another feature descriptor based upon SIFT is Affine-SIFT (ASIFT) proposed by Morel and Yu (2009). ASHIFT follows affine transformation parameters to correct images and is intended to resist strong affine issues.

Maximally stable extremal regions (MSER) The blob detection method Maximally stable extremal regions was developed by Matas et al. (2004). MSER regions are areas of connected pixels which are characterized by an almost uniform intensity and are surrounded by contrasting backgrounds. These regions are found by applying a multitude of thresholds over the image. Those regions of which the shape remains unchanged over a large set of thresholds are selected as MSER regions. A key property of MSER is that it is invariant to affine transformation of image intensities.

Pros and cons different algorithms Juan and Gwun (2009) compared the performance of SIFT, PCA-SIFT and SURF on deformed images. They concluded that there is no method which performs best on all transformations. The results of Juan & Gwun are summarized in Table A.2.

Method	Time	Scale	Rotation	Blur	Illumination	Affine
SIFT	Common	Best	Best	Best	Common	Good
PCA-SIFT	Good	Common	Good	Common	Good	Good
SURF	Best	Good	Common	Good	Best	Good

TABLE A.2: Results from Wu et al. (2013) of different SIFT variations on images with different transforms.

Wu et al. (2013) also compared the performance of SIFT, PCA-SIFT and SURF on deformed images. In addition they also tested the SIFT variants GSIFT, CSIFT and ASIFT. Their results underline the findings of Juan & Gwun, each algorithm has its own applications in which it performs best. It is therefore import to choose the correct algorithm depending on the type of deformation of the image.

The results show that SIFT and CSIFT have the best performance under scale and rotation changes. CSIFT outperforms CSIFT under blur and affine changes, but not under illumination changes. ASIFT has the best performance under affine changes, while GSIFT has the best performance under blur and illumination changes. PCA-SIFT always the second best in different situations. SURF has the fastest performance, but performs the worst in different situations.

Method	Time	Scale &Rotation	Blur	Illumination	Affine
SIFT	Better	Best	Good	Better	Good
PCA-SIFT	Better	Better	Better	Better	Good
SURF	Best	Common	Common	Common	Common
GSIFT	Better	Good	Best	Best	Good
CSIFT	Good	Best	Better	Good	Better
ASIFT	Common	Good	Common	Common	Best

TABLE A.3: Results from Wu et al. (2013) of different SIFT variations on images with different transforms.

A.5.1 Classification

After describing and extracting regions of interest, it is often useful to classify and label those extracted regions. Often techniques borrowed from Machine Learning are used for the classification.

A.5.1.1 k-Nearest Neighbour

The k-Nearest Neighbour algorithm is a non-parametric method which can be used to classify data into a number of classes. Basically, an object is classified according to the majority vote of its k closest neighbours. Often a weighing is added to make the vote stringer of neighbours that are the closest to the object. Determining which objects are the closest is done with distance measures such as the Euclidean distance. One of the downsides of k-Nearest Neighbours is that it is relatively sensitive to local structures in the data.

A.5.1.2 Artificial Neural Networks

Artificial Neural Networks is a well-known method for the classification of data, such as visual data. Artificial Neural Networks aim at simulating the neural networks of our brain at a very basic level. Within the brain, neurons are interconnected with each other. They send out signals to each other called synapses. When the sum of all incoming synapses for a node exceeds a certain threshold, that node will 'fire' and send synapses to its connected nodes.

A artificial neural network emulates this behaviour. The neurons of the brain are simulated with nodes that are connected to each other. Incoming signals from the other nodes are summarized and a certain activation function is used to determine if the node should fire.

A.5.1.3 Decision Tree learning

Decision trees are tree-like graphs or models of decisions and the consequences of these decisions. Within decision tree learning, these trees are used to map observations about an item to conclusion about the items target value. Leaves of the trees represent class labels, while branches represent the conjunctions of features that lead to the class labels.

A.5.1.4 Support Vector Machines

Support Vector Machines (SVMs) (Cortes and Vapnik, 1995) can be used to classify data, such as visual data, into two classes. Based on a set of labelled training data, SVM tries to find a pair of parallel hyperplanes that lead to a maximum separation between the two classes. If the data is not linearly separable, SVM transforms the training and test data to a higher dimension in which non-linear classification with a kernel function is performed.

A.5.1.5 Boosting

The technique of boosting was developed by Schapire (1990). The basic idea behind boosting, is combining multiple weaker classifiers to create one strong classifier. New weak classifiers are iteratively learned and added to the final strong classifier. These weak classifiers are often weighted based upon their accuracy. In order to let the new classifiers mainly focus upon the incorrectly classified data, the data is reweighed after the addition of each classifier. Misclassified examples gain weight, while correctly classified examples lose weight.

A multitude of boosting algorithms have been developed in the last couple of decades. One of the most well-known boosting method, is the method of Adaptive Boost (AdaBoost) developed by (Freund and Schapire, 1995). ?

A.6 Libraries, Frameworks and Toolboxes

Over the years, a large variety of libraries, toolboxes and frameworks have been developed to aid the use of computer vision for a wide variety of applications. In this section, an overview of the most well used software package (libraries, toolboxes and frameworks) for computer vision will be given. Furthermore, the functionality of these software packages will be compared in a decision matrix.

A.6.1 Libraries

OpenCV OpenCV, which is short for Open Source Computer Vision, is a computer vision library originally developed by Intel and which is now supported by Willow Garage and Itseez. It is a platform independent and open-source library which is released under a BSD license which makes it free to use for both commercial and academic use. It has been written in optimized C/C++ and is able to take advantage of multi-core processors.

VLFeat VLFeat is a different open-source library aimed a computer vision. It is written in C, and is able to interface with MATLAB and Octave. It is released under de BSD license which makes It free to use for commercial and academic applications.

A.6.2 Toolboxes

Matlab image Processing Toolbox (IPT) The Matlab Image Processing toolbox which has been developed by Mathworks, provides a wide range of functions and apps for image processing tasks such as image analysis, image segmentation, image enhancement, image registration and noise reduction.

Matlab Computer Vision System Toolbox (CVST) The Matlab Computer Vision System toolbox, developed by Mathworks, delivers a variation of functions and apps for computer vision systems. These include object detection and tracking, feature detection and extraction, feature matching and stereo vision.

Matlab Machine Vision Toolbox(MVT) The Machine vision toolbox for Matlab was developed by Corke (2007). It contains a wide range of functions that are useful in both machine vision as well as computer vision.

A.6.3 Frameworks

SimpleCV SimpleCV, which stand for Simple Computer Vision, is a computer vision framework aimed at making computer vision easy (Demaagd, Oliver, Oostendorp & Scott 2012). It bundles together open-source computer vision libraries such as, the earlier discussed, OpenCV and algorithms to solve computer vision problems. It is written in Python and released under a BSD license.

AForge.NET Aforge.NET is computer vision and Artificial Intelligence framework written in C#. It is open-source and available under the Lesser GPL license which makes it free for both commercial and academic use. Overview

A.6.3.1 Overview

An overview of the difference licence options and API's for the discussed computer vision software package is given in Table A.4.

	API	Licence
OpenCV	C++, C#, Java, Python	BSD
VLFeat	Matlab/Octave, C	BSD
Matlab IPT	Matlab	Commercial
Matlab CVST	Matlab	Commercial
Matlab MVT	Matlab	Lesser GPL
SimpleCV	Python	BSD
Aforge.NET	C#	Lesser GPL

TABLE A.4: Overview of the different API's and licences offered by the different software packages.

A.6.4 Decision Matrix

Continues on the next page.

	OpenCV	**VLFeat**	**Matlab IPT**	**Matlab CVST**	**Matlab MVT**	**Simple CV**	**Aforge .Net**
Noise removal							
Linear Filter	x		x				
Median Filter	x		x			x	x
Gaussian Smoothing	x		x		x	x	x
Contrast enhancement							
Histogram Scaling							
Histogram Equalization	x		x			x	x
Feature detection							
Prewitt	x		x			x	
Sobel	x		x		x		x
Laplacian	x				x		
LOG			x		x		
Canny	x		x		x		x
Isodata thresholding	x						x
K-means clustering	x	x	x		x		
Growing-and-merging							
Splitting-and-merging	x						
Watershed	x		x			x	
Corner detection							
Moravec Corner Detection							x
Harris & Stephens	x		x		x		
Blob detection							
Difference of Gaussian					x		
Feature detectors							
SIFT	x	x			x	x	
PCA-SIFT							
GLOH							
SURF	x			x		x	
GSIFT							
CSIFT							
ASIFT							
MSER	x	x		x	x	x	
Classification							
K-nearest neighbours classification	x	x				x	
Artificial Neural Networks	x						x
Decision Tree Learning	x					x	
Support Vector Machine	x	x				x	
Boosting	x						

A.6.5 Conclusion

The Decision matrix gives an overview of the availability of the techniques discussed in section.. within each library/framework/toolbox. As can be seen from this matrix, all libraries, frameworks and toolboxes offer a different set of implemented computer vision algorithms. Depending on the requirements of the project, the decision matrix can be used to find the most suitable library, framework or toolbox. Although it can be important to also take into consideration the license (and price of the license) and programming language that can be used.

Overall, the matrix shows that OpenCV offers the most complete package of implemented computer vision algorithm. Furthermore, since it is offered under a BSD licence it is free to use for researchers and businesses for their own software. Another advantage of the BSD license is that, unlike the GPL license, it does not require the build software to be released under the BSD license.

The toolboxes offered by Mathworks complement each other to offer a broad selection of computer vision algorithms. Additional classification algorithms can be found in other toolboxes offered by Mathworks such as the Statistics Toolbox, Neural Network Toolbox and Fuzzy Logic Toolbox. The downside of using the Mathworks toolboxes is the relatively high licensing costs for each toolbox, and the Matlab environment.

The Matlab Machine vision toolbox, offers a broad range of computer vision algorithms including widely used feature detection algorithms. However, it does not offer any classification algorithms. The Matlab Machine vision toolbox is released under the Lesser GPL license. The lesser GPL license requires that any improvement of the Matlab Machine vision toolbox code by the user, has to be made publicly available. Though, unlike the GPL license, it does not require for the source of any software which is only linked to the Matlab Machine vision toolbox framework to be publicly released as open-source.

Since SimpleCV is based upon OpenCV it has the same license advantages as OpenCV. Furthermore, it also offers a broad variety of computer vision algorithms, which are, according to the creators, easier to use compared to OpenCV and other packages.

Aforge.NET offers an implementation of most basic computer vision algorithms, but offers only Neural Networks for classification purposes. Furthermore it does not offer any of the discussed feature detection algorithms. Like the Matlab Machine vision toolbox, Aforge.net is also released under the Lesser GPL license.

VLFeat strength lies mainly in its feature detection and classification algorithms. It does not offer algorithms for basic computer vision task such as noise removal. Like OpenCV

and SimpleCV, it license gives its user the freedom to use the framework without the need to pay, or make their own code open-source.

Appendix B

Plots

B.1 Smartphone

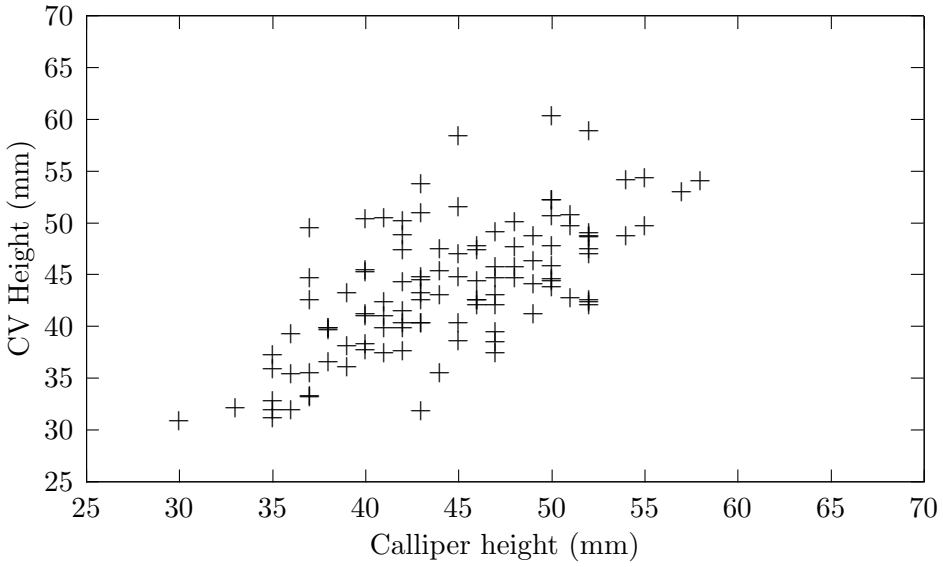


FIGURE B.1: Height measurements by calliper versus height measurements by the computer vision (CV) system for the single potato condition based on images taken with the Xiaomi RedMi

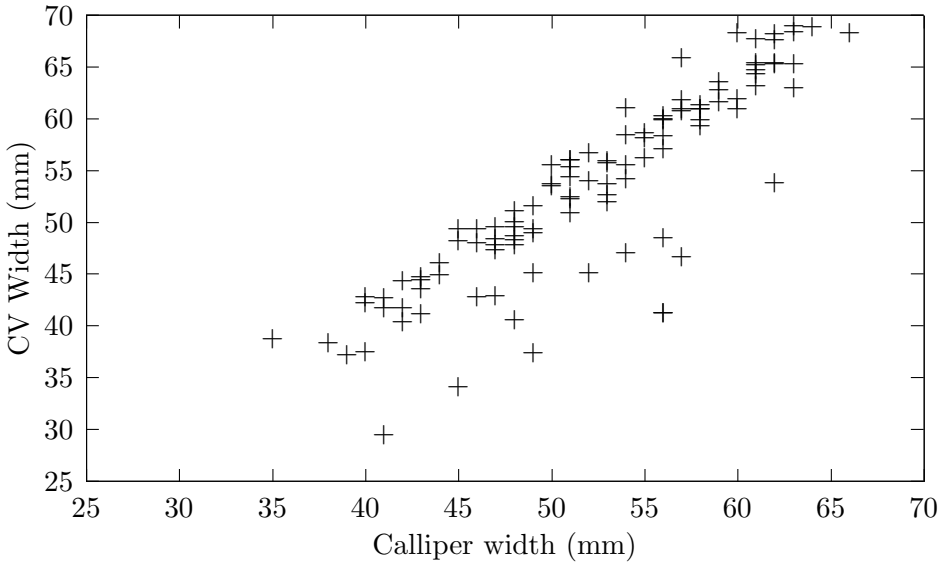


FIGURE B.2: Width measurements by calliper versus width measurements by the computer vision (CV) system for the single potato condition based on images taken with the Xiaomi RedMi

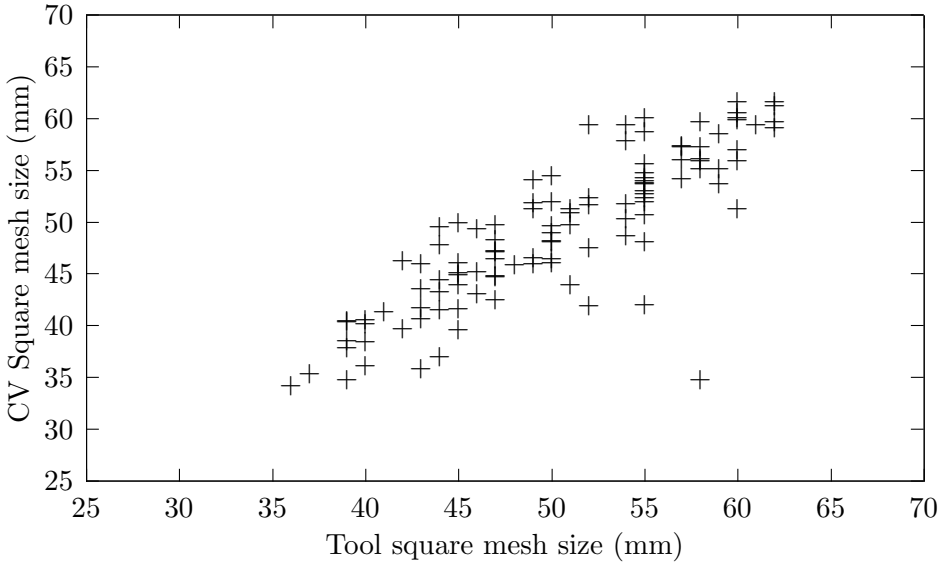


FIGURE B.3: Square mesh size measurements with the square mesh size tool versus square mesh measurements by the computer vision (CV) system for the single potato condition based on images taken with the Xiaomi RedMi

B.2 Google Glass

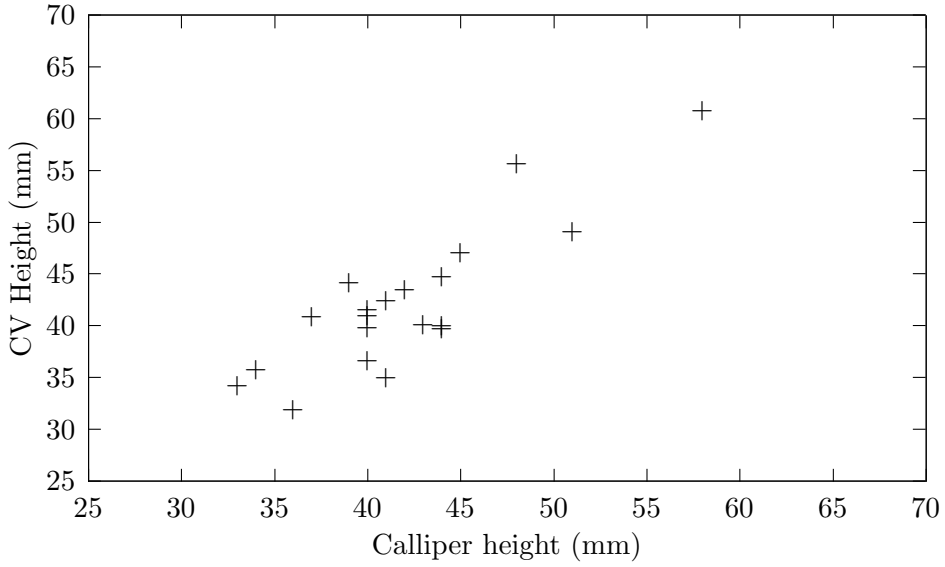


FIGURE B.4: Height measurements by calliper versus height measurements by the computer vision (CV) system for the single potato condition based on images taken with the Google Glass

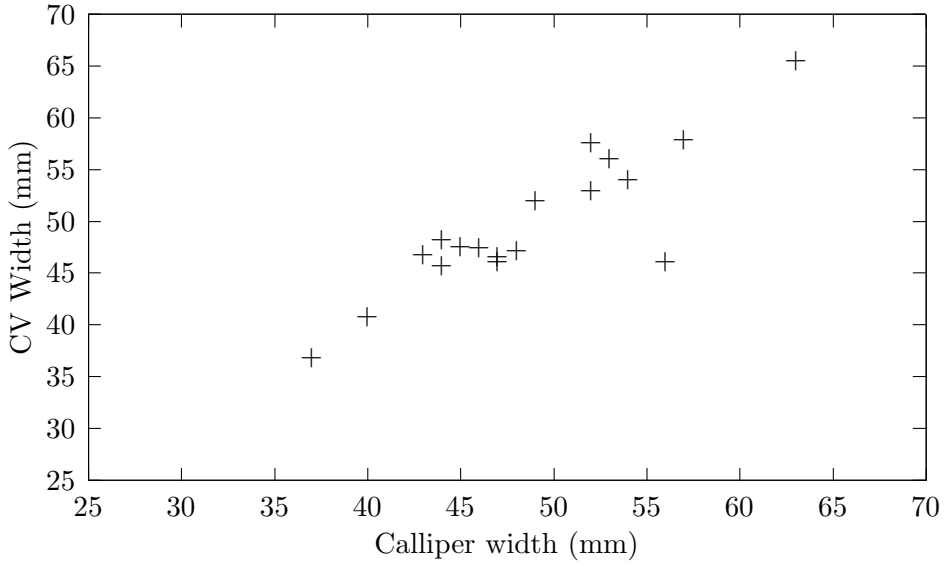


FIGURE B.5: Width measurements by calliper versus width measurements by the computer vision (CV) system for the single potato condition based on images taken with the Google Glass

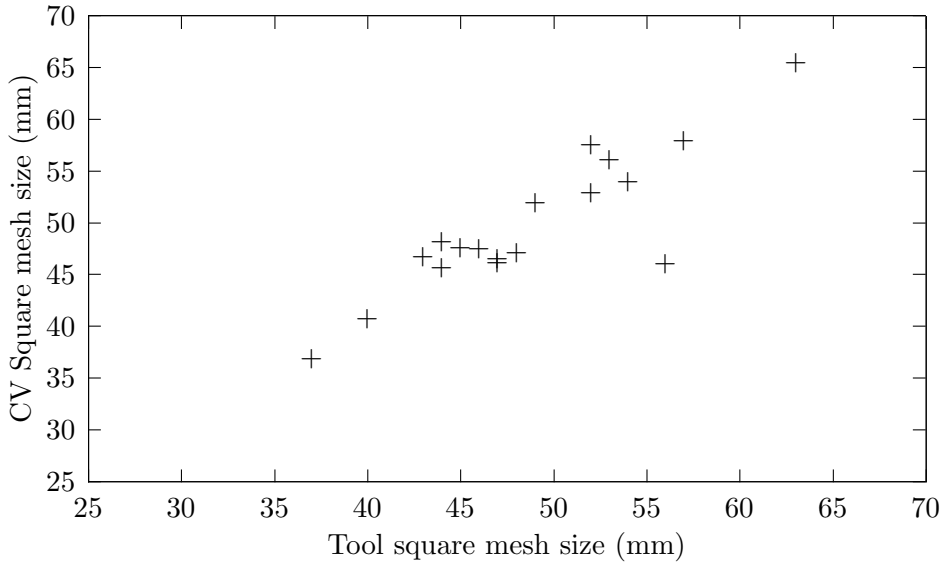


FIGURE B.6: Square mesh size measurements with the square mesh size tool versus square mesh measurements by the computer vision (CV) system for the single potato condition based on images taken with the Google Glass

B.3 Multiple potatoes

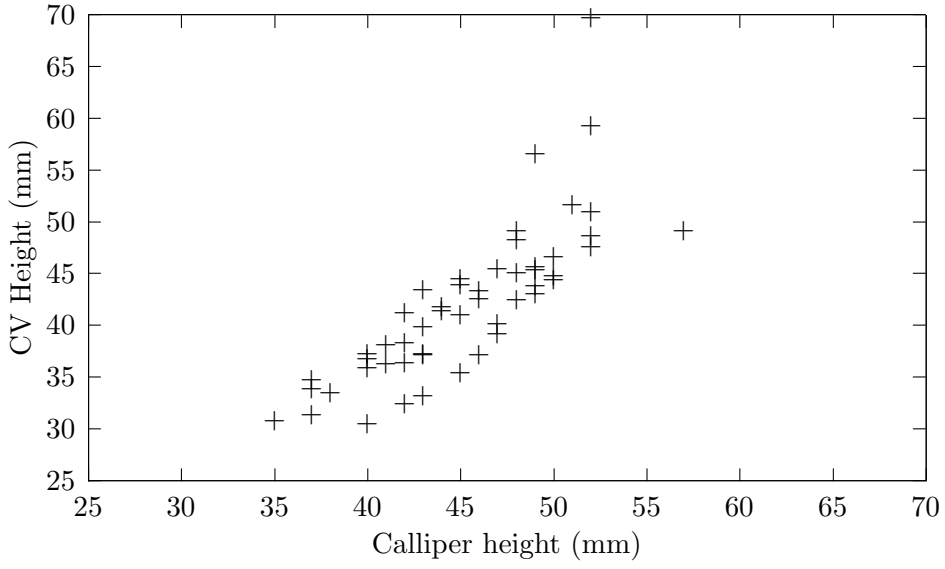


FIGURE B.7: Height measurements by calliper versus height measurements by the computer vision (CV) system for the multiple potato condition based on images taken with the Xiaomi RedMi

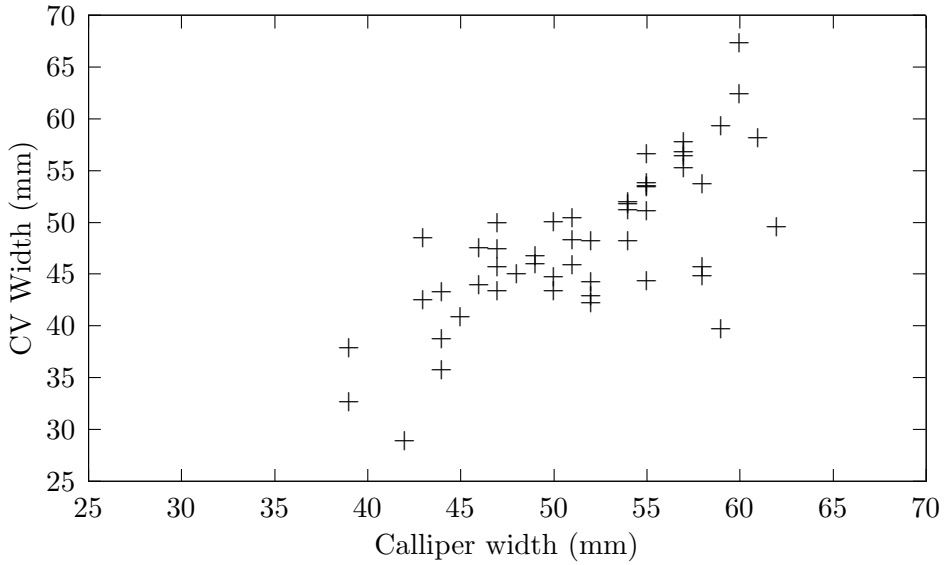


FIGURE B.8: Width measurements by calliper versus width measurements by the computer vision (CV) system for the multiple potato condition based on images taken with the Xiaomi RedMi

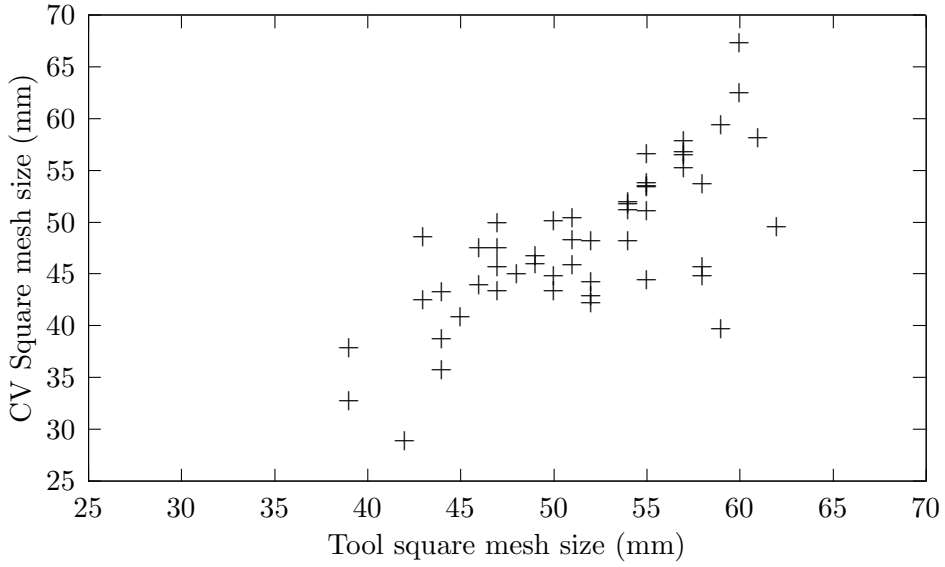


FIGURE B.9: Square mesh size measurements with the square mesh size tool versus square mesh size measurements by the computer vision (CV) system for the multiple potato condition based on images taken with the Xiaomi RedMi

Appendix C

Data

C.1 Xiaomi RedMi

ID	Calliper Height	Calliper Width	Tool Square mesh size	CV Height	CV Width	CV Square mesh size
X1	39	49	45	38.08	49.03	43.90
X2	37	48	44	33.34	48.36	41.53
X3	36	41	40	39.34	41.77	40.57
X4	37	44	39	35.56	44.96	40.54
X5	30	39	36	30.91	37.21	34.21
X6	44	53	50	43.12	52.62	48.11
X7	40	35	39	38.28	38.76	38.52
X8	41	52	47	37.41	54.07	46.49
X9	43	54	50	44.77	54.17	49.69
X10	38	43	43	39.83	43.61	41.76
X11	36	43	40	35.47	41.18	38.43
X12	41	49	47	39.92	49.32	44.87
X13	50	63	58	47.77	62.99	55.90
X14	35	42	39	35.95	44.38	40.39
X15	38	41	41	39.83	42.74	41.31
X16	42	46	45	41.48	48.06	44.89
X17	35	42	40	31.20	40.41	36.10
X18	33	38	37	32.17	38.37	35.41
X19	45	51	49	38.67	52.33	46.01
X20	38	48	44	36.63	51.17	44.50
X21	39	43	43	36.05	44.77	40.64

ID	Calliper Height	Calliper Width	Tool Square mesh size	CV Height	CV Width	CV Square mesh size
X22	52	64	62	48.74	68.90	59.68
X23	47	58	55	44.67	61.34	53.66
X24	46	54	52	47.39	55.62	51.67
X25	46	58	55	47.77	60.95	54.76
X26	58	61	62	54.07	67.74	61.29
X27	35	40	40	37.31	42.83	40.17
X28	39	50	47	43.22	55.62	49.81
X29	47	61	58	49.13	64.34	57.25
X30	46	53	54	44.38	55.72	50.37
X31	51	63	59	50.78	65.31	58.50
X32	36	44	42	31.98	46.13	39.69
X33	47	57	55	42.06	60.95	52.36
X34	54	65	60	48.74	70.55	60.63
X35	43	51	49	53.78	54.36	54.07
X36	45	54	50	47.00	61.05	54.48
X37	40	45	44	37.79	48.16	43.29
X38	54	66	60	54.17	68.32	61.65
X39	37	43	43	42.64	44.48	43.57
X40	51	63	60	49.71	69.00	60.13
X41	43	46	45	42.64	49.32	46.10
X42	45	55	50	40.31	56.30	48.96
X43	48	50	50	50.10	53.69	51.92
X44	52	54	55	49.03	58.53	53.99
X45	41	45	45	50.49	49.42	49.96
X46	50	57	55	45.84	60.76	53.82
X47	47	56	55	39.54	59.89	50.74
X48	44	59	55	47.48	62.79	55.67
X49	50	60	57	52.23	61.92	57.28
X50	52	55	55	47.48	58.14	53.08
X51	50	62	60	44.58	65.41	55.97
X52	40	48	45	41.09	48.74	45.08
X53	47	53	54	43.03	53.69	48.65
X54	43	46	45	40.41	42.83	41.64
X55	42	51	49	47.39	56.01	51.88
X56	50	64	60	44.38	72.10	59.87
X57	52	55	55	58.92	58.63	58.77

ID	Calliper Height	Calliper Width	Tool Square mesh size	CV Height	CV Width	CV Square mesh size
X58	55	57	58	49.71	61.83	56.10
X59	42	47	46	50.20	48.45	49.33
X60	55	62	62	54.36	68.22	61.68
X61	48	61	57	45.74	64.73	56.05
X62	49	60	57	46.42	60.95	54.17
X63	52	61	59	42.06	63.18	53.67
X64	50	58	55	60.37	59.89	60.13
X65	42	47	46	37.70	47.87	43.08
X66	49	56	54	44.19	58.34	51.75
X67	47	63	58	37.41	68.41	55.13
X68	57	61	61	53.01	65.22	59.43
X69	46	51	51	42.54	56.11	49.79
X70	47	51	50	38.52	52.52	46.06
X71	45	59	54	51.55	63.57	57.87
X72	37	48	44	49.52	49.52	49.52
X73	52	62	60	48.65	53.88	51.33
X74	35	40	39	32.85	42.25	37.84
X75	46	59	55	42.06	61.63	52.76
X76	44	56	52	35.56	57.08	47.55
X77	48	56	55	47.68	48.55	48.12
X78	38	45	44	39.73	34.11	37.03
X79	47	54	50	45.74	47.10	46.42
X80	40	41	43	41.18	29.46	35.81
X81	42	52	47	44.29	45.16	44.72
X82	51	56	55	42.74	41.28	42.01
X83	46	56	52	42.64	41.28	41.97
X84	44	53	51	45.35	55.91	50.91
X85	52	62	60	47.10	65.31	56.94
X86	43	49	58	31.88	37.41	34.75
X87	43	47	47	50.97	42.93	47.12
X88	42	51	50	39.92	55.33	48.25
X89	50	60	57	43.80	68.32	57.38
X90	43	51	48	40.31	50.88	45.90
X91	49	58	55	41.18	60.95	52.02
X92	41	57	51	40.99	46.71	43.94
X93	48	49	47	44.67	51.65	48.29

ID	Calliper Height	Calliper Width	Tool Square mesh size	CV Height	CV Width	CV Square mesh size
X94	42	53	49	40.31	52.04	46.55
X95	40	52	49	45.25	56.69	51.29
X96	45	56	52	58.43	60.27	59.36
X97	50	62	58	50.68	67.64	59.76
X98	50	57	54	52.23	65.90	59.46
X99	52	67	62	42.64	71.90	59.11
X100	49	58	55	48.74	59.31	54.28
X101	43	56	52	43.32	59.98	52.32
X102	52	61	59	42.44	65.41	55.14
X103	40	48	44	45.45	50.06	47.81
X104	41	48	46	42.44	47.87	45.24
X105	45	47	47	44.87	49.52	47.25
X106	35	40	39	31.98	37.50	34.85
X107	37	49	45	33.24	45.06	39.59
X108	37	47	43	44.67	47.29	46.00
X109	42	50	51	48.84	53.59	51.27
X110	43	48	47	44.48	40.60	42.59
X111	40	42	42	50.39	41.77	46.28

TABLE C.1: Data from the hand measurements and the computer vision measurements for the Xiaomi RedMi dataset.

C.2 Google Glass

ID	Calliper Height	Calliper Width	Tool Square mesh size	CV Height	CV Width	CV Square mesh size
G1	37	46	43	40.89	49.32	45.31
G2	44	56	52	44.77	57.08	51.29
G3	40	48	45	39.83	51.65	46.12
G4	48	53	52	55.62	55.91	55.77
G5	44	46	56	39.73	49.03	44.63
G6	44	60	54	40.02	62.21	52.31
G7	40	51	47	41.48	47.68	44.68
G8	40	46	44	36.63	50.78	44.27
G9	51	60	57	49.03	62.31	56.07
G10	42	51	49	43.51	56.30	50.31
G11	58	64	63	60.76	66.09	63.48
G12	45	56	53	47.00	60.86	54.37
G13	43	49	47	40.12	49.62	45.12
G14	41	52	48	34.98	54.27	45.65
G15	40	48	45	40.99		
G16	39	47	44	44.19	49.03	46.67
G17	34	43	40	35.76	42.93	39.51
G18	41	49	46	42.35	49.42	46.02
G19	36	40	39	31.88		
G20	33	40	37	34.21	37.21	35.74

Table C.2: Data from the hand measurements and the computer vision measurements for the Google Glass dataset.

C.3 Multiple potatoes

ID	Calliper Height	Calliper Width	Tool Square mesh size	CV Height	CV Width	CV Square mesh size
M1	48	61	57	42.44	65.70	55.31
M2	49	60	57	45.35	65.70	56.45
M3	52	61	59	50.97	66.77	59.40
M4	50	58	55		62.02	
M5	42	47	46	41.18	53.20	47.57
M6	49	56	54	43.80	57.66	51.20
M7	47	63	58	39.15	65.12	53.73
M8	57	61	61	49.13	65.90	58.12
M9	46	51	51	37.11	53.20	45.87
M10	47	51	50	40.12	46.51	43.43
M11	45	59	54	35.47	64.44	52.01
M12	37	48	44	34.79	36.73	35.77
M13	52	62	60	59.31	65.51	62.48
M14	35	40	39	19.67	41.86	32.71
M15	46	59	55	42.54	63.09	53.80
M16	44	56	52	41.38	54.17	48.20
M17	48	56	55	48.26	58.34	53.54
M18	38	45	44	33.53	43.41	38.79
M19	47	54	50	45.45	54.36	50.10
M20	40	41	43	35.85	58.53	48.54
M21	42	52	47	36.44	60.57	49.98
M22	51	56	55	51.65	50.49	51.07
M23	46	56	52	43.32	45.06	44.20
M24	44	53	51	41.77	57.76	50.40
M25	52	62	60	69.67	64.93	67.34
M26	43	49	58	37.11	51.46	44.86
M27	43	47	47	33.14	51.65	43.39
M28	42	51	50	32.46	54.36	44.77
M29	50	60	57	44.77	68.41	57.81
M30	43	51	48	37.31	51.65	45.05
M31	49	58	55	45.64	60.27	53.46
M32	41	57	51	38.18	56.69	48.33
M33	48	49	47	45.06	49.81	47.49
M34	42	53	49	38.37	52.52	46.00

ID	Calliper Height	Calliper Width	Tool Square mesh size	CV Height	CV Width	CV Square mesh size
M35	40	52	49	36.82	54.95	46.77
M36	45	56	52	43.90		
M37	50	62	58	44.38	47.00	45.71
M38	50	57	54	46.61	49.81	48.24
M39	52	67	62	47.58	51.55	49.61
M40	49	58	55	43.03	45.74	44.40
M41	43	56	52	39.83	44.48	42.22
M42	52	61	59	48.65	28.01	39.69
M43	40	48	44	37.21	48.65	43.31
M44	41	48	46	36.34	50.49	43.99
M45	45	47	47	41.09	49.91	45.71
M46	35	40	39	30.82	43.80	37.87
M47	37	49	45	31.40	48.55	40.88
M48	37	47	43	33.92	49.71	42.55
M49	48	61	57	49.13	63.57	56.81
M50	45	59	54	44.48	58.14	51.76
M51	42	52	47		44.87	
M52	49	58	55	56.50	56.69	56.59
M53	43	56	52	43.51	42.35	42.93
M54	40	42	42	30.53	27.13	28.88

TABLE C.3: Data from the hand measurements and the computer vision measurements for the multiple potatoes dataset.