TSB-funded Project 'TADD' - Trainable vision-based anomaly detection and diagnosis

Ran Song, Hossein Malekmohamadi and Tom Duckett

Agri-Food Technology Research Group, Lincoln Centre for Autonomous Systems Research, School of Computer Science

University of Lincoln, UK

Technical Quarterly Report

Feb 2014

Contents

List of Figures

Chapter 1

Introduction

This technical quarterly report is a summary of the work we have done in November, December 2013 and January 2014 for the project Trainable Vision-based Anomaly Detection and Diagnosis (TADD). We integrate the corresponding six monthly technical reports as Chapters 2-7. Chapters 2-4 are Dr Ran Song's reports and Chapters 5-7 are Dr Hossein Malekmohamadi's reports.

Chapter 2

Ran Song TADD Technical Report for Nov 2013

2.1 Abstract

In November, we investigate the techniques which can be used for label recognition on food tray. The techniques have to be efficient and reliable because we pursue a real-time system capable of handling input food tray images of high variety. We finally ended up with feature-based image registration techniques using SURF and RANSAC. Using such techniques, the system is able to answer three questions related to label recognition: (1) is some label on the tray? (2) if it is on the tray, is it at the right position? (3) if it is not at the right position, how wrong is its position?

2.2 Introduction

In this report, we give details about the techniques we used for label recognition on food tray images. The basic idea is to utilise SURF (Speeded Up Robust Features) Bay *et al.* [2006] and RANSAC (Random Sample Consensus) Fischler & Bolles [1981] to register a test image into the coordinate system of a reference image where everything is assumed to be right. In the current TADD system, we have also moved to SURF from SIFT Lowe [2004] for feature detection due to

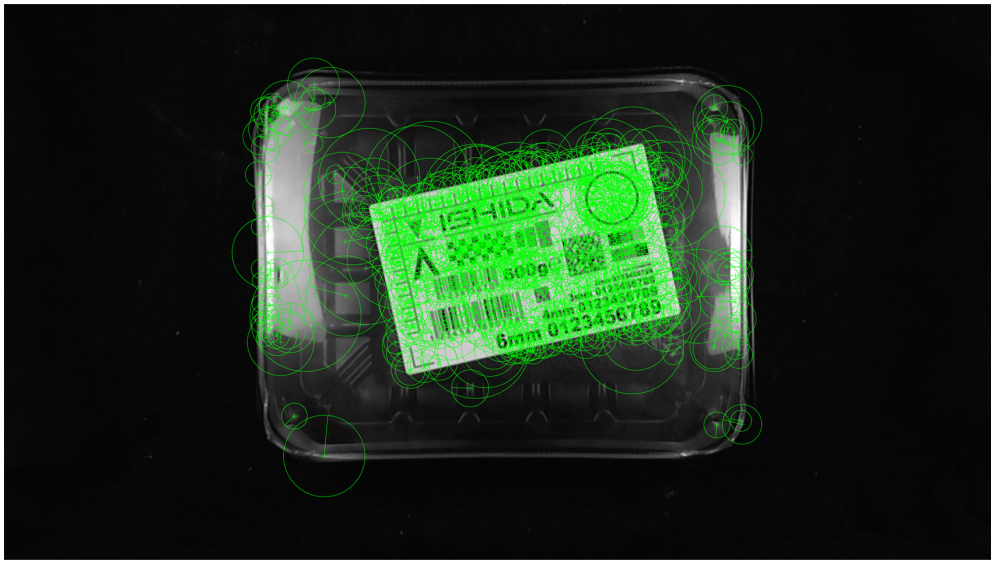


Figure 2.1: 913 SURF features are detected on a food tray image. The '+' denotes the location of each feature, the size of the circle around each '+' denotes the scale of each feature and the line segment within the circle denotes the orientation of the feature.

efficiency reason.

Figure 2.2: The strongest 100 SURF features out of the 913 features shown in Figure. 2.1

Figure 2.3: The raw matches produced by SURF contains some incorrect matches. To visualise the results, we deliberately colour the two images in red and blue respectively.

2.3 Methods and results

The SURF method is technically complicated for non-expert. Thus here we just give a simple explanation on how it works. SURF contains two components: a detector and a descriptor. A detector is to find the locations of the features, which thus allows us to visualise them as points in an image. A descriptor is to dig out the (mostly local) properties attached to the points which can be used to distinguish them and match them over different images in the presence of image rotation, translation, scale change and illumination change, etc.

The detector is based on the Hessian matrix, but uses a very basic approximation. Such an approximation relies on integral images to reduce the computation time. The descriptor, on the other hand, describes a distribution of Haar-wavelet responses within the neighbourhood of the feature point. Again, integral images are exploited for speed. Figure. 2.1 illustrates our results of applying SURF detector over food tray images and Figure. 2.2 shows the strongest 100 SURF features.

Although the SURF method offers a descriptor to facilitate the matching

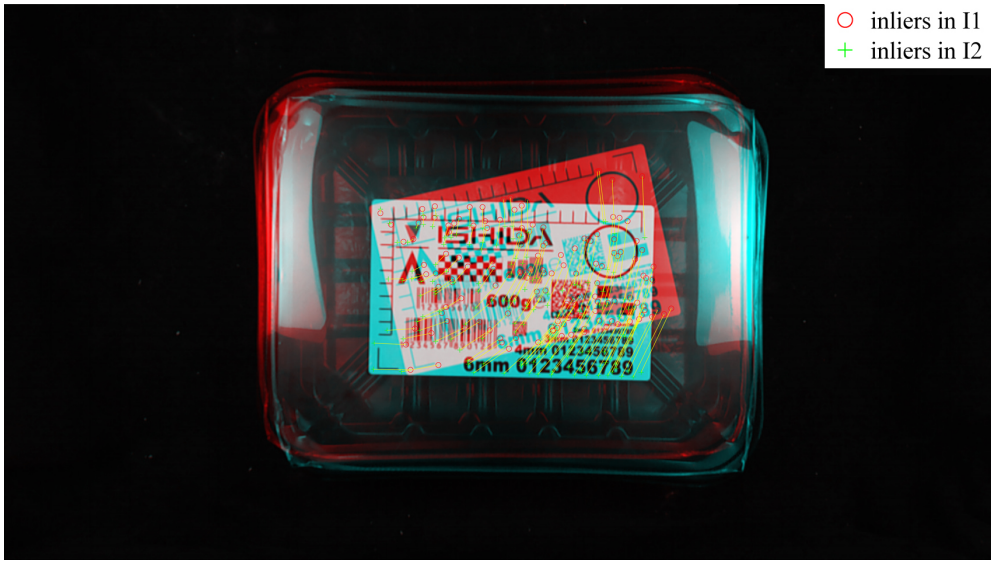


Figure 2.4: By using RANSAC, the matches are refined. In this case, only the feature points within the label region are preserved.

of the features and indeed it achieves a high matching rate of 82.6% compared with 78.1% of SIFT, it cannot eliminate mismatches which could lead to wrong registration in the following steps. Therefore, we introduce RANSAC to refine the matches produced by the original SURF and further eliminate all mismatched feature points.

The basic theory of the RANSAC method is simple but ingenious. Firstly, it randomly samples two feature points (from the set of matched feature points produced by SURF) to fix one line (which represents the least-square solution of the motion estimation between two images in our application). The underlay (or consensus) of this line is defined as the points whose distance to this line are less than some threshold. After many iterations of this random sampling approach, the final solution is the line which has the largest consensus and the points within the consensus are the inliers that consist of the consensus. The rest of the points are viewed as outliers. To ensure that the random sampling has a good chance of finding a true set of inliers, a sufficient number of samplings must be tried. In this application, we set it to 2000. Figure. 2.3 shows the result of raw feature matching produced by SURF where we use the yellow lines to illustrate the matches. It can be seen that some features points are not correctly matched.

Figure 2.5: The two input images where the bottom one is the reference image.

A correct match means that the two matched points should correspond to the same point in the real world although they are from different images. In contrast, Figure. 2.4 demonstrates that by using the RANSAC method, we can eliminate the incorrect matches where each yellow line correctly links two features which correspond to the same point the real world.

Once we have produced a set of correctly matched feature points, we can use them to do registration. As shown in Figure. 2.4, all of the inliers are within the label region. Hence, we actually register one label to the other reference label which is supposed to be placed correctly.

Figure 2.6: Rectified test image where the label is actually at the same position as that in the reference image

Rigid image registration is a fundamental topic in computer vision. In this work, we assume that the motion between two images are affine transform. Such a motion can be formulated as a set of linear equations given a known set of matched points. The unknown parameters, the 6 motion parameters (which describe the rotation and translation between the two labels in this case) can be solved in a least square manner using for example, Levenberg-Marquardt method. Thus by computing the motion parameters, we can know the relative position and orientation of the label to the reference one. In other words, if we assume that the label is at the right position with the right orientation in a reference food tray image, we can know how wrong the position and the orientation of the label in a test image are by doing registration. Figure. 3.1 shows one test image and one reference image and Figure. 2.6 shows the new position and orientation of the test image after the registration. Figure. 3.5 displays the overlapped feature points in the two images. It can be observed that each pair of corresponding feature points from different images have already been moved to the same position after the registration.

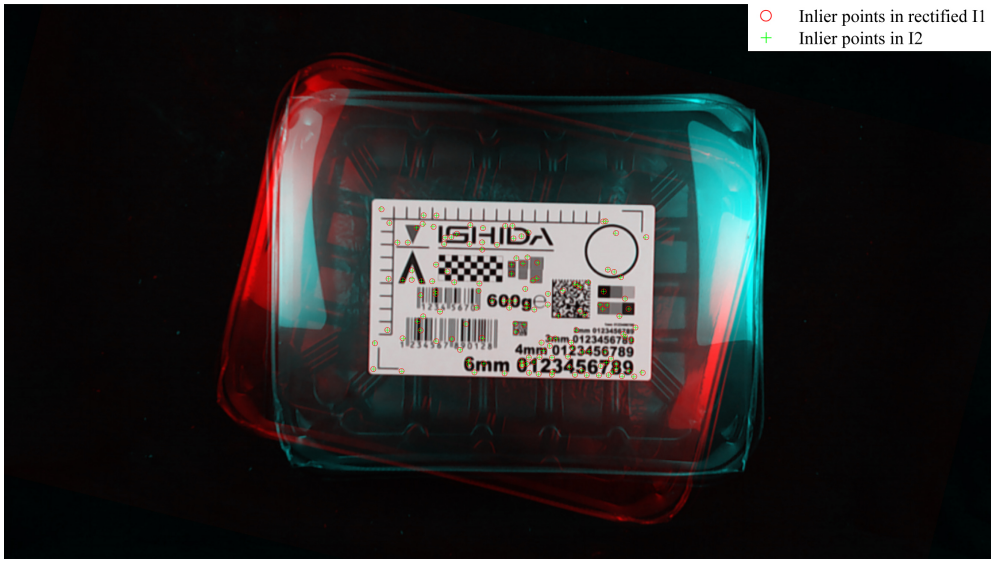


Figure 2.7: After registration, the feature points from different images are fully overlapped.

2.4 Conclusions and future work

The techniques reported above demonstrated the capability of combined 'SURF+RANSAC' solution over the issue of label recognition. According to our preliminary implementation without any code optimisation, the whole process needs about 2.5 seconds. In short, 'SURF+RANSAC' can answer the three questions that we raised at the beginning of this report as long as we know the reference image or simply 'what is the right one like'. And this will guide us in our future work: how can we make the system learn 'what is the right one like'?

The original idea is to first input a tray with blank film and then input another tray with only one label at the right position. Then we need to implement SURF method on both images to output two sets of features. The difference set between the two sets is supposed to be the set of SURF features within the label region. However, our tests show that due to the strong reflection, this is not always the case. Reflection can produce new features on a food tray image. In fact, in Figure. 2.1, most of the detected SURF features outside the label region are not driven by the pattern and the texture of the tray itself but caused by the reflection. The key thing is that there is no guarantee that we can always lay

trays at the same position so that the reflections on the surface of the trays always change. And that will lead to different results of SURF feature detection in the same region of a tray. However, because we are currently not quite sure about the eventual lighting condition, we will revisit this idea by using input images captured under different lighting conditions (e.g., using the vision system attached to the Ishida machine).

A potentially good solution to this problem is to do the graph-based segmentation within the tray region. As we mentioned in our previous reports, we used graph-based method for foreground-background segmentation. We may extend this method in order to further partition the foreground region. First we input a reference tray image where all of labels are positioned correctly and apply SURF on it. If the graph-based method can reliably partition its foreground tray region into different components, we can record the SURF features locating in each component. And as long as we can do a semantic labelling of the components, the system will know what some component is (e.g., a price label or a barcode label). In this way, the system can learn which SURF features correspond to a specific label. Given that a set of SURF features corresponding to the price label are detected in the reference image, we can check whether there are a enough number of matches detected in a test tray image. If the answer is 'No', it means that the price label is missing. If the answer is 'Yes', we can further figure out whether the label is at the right position and how wrong its position is using the aforementioned registration-based methods. And this scheme could be more desirable since we do not need to input trays again and again if one tray contains several labels.

Chapter 3

Ran Song TADD Technical Report for Dec 2013

3.1 Abstract

In this report, we proposed an automatic method in order to detect the region of the label within a tray image. This method is based on some classic computer vision techniques, including feature extraction, image registration, image erosion and dilation. Such computer vision techniques are specifically integrated, forming an effective object detection system which can handle various images of food trays with different positions and orientations. Preliminary experiments show that the proposed method is efficient, reliable and easy-to-use (in particular for non-expert) while certainly more tests need to be done to comprehensively evaluate it.

3.2 Introduction

The automatic detection of the label region on a food tray is very important in modern food industry. On the one hand, we need to make sure that the label, usually containing such information as barcode, dates, price and weight is presented at the right position and orientation. On the other hand, the detection of the label region is typically a necessary preprocessing step for the Optical Character

Recognition (OCR) so that the image information of the printed texts within the label region can be converted into machine-encoded/computer-readable text and then the system can judge whether the printed dates, price and weight are semantically right or wrong. Importantly, the accuracy of the label region detection largely affects the reliability of the following OCR.

However, detecting the label region from a complicated image of food tray is not a simple task, which usually consists of some advanced computer vision and/or pattern recognition techniques due to the complex and various content of the input images Fang & Xie [2010]. If we further consider whether the detection is fast enough and the system is easy to operate (both are vital for industrial applications), the task will be more challenging. In this report, we propose a novel method to solve the problem.

3.3 Method

The proposed method is basically a teaching-by-showing approach. We first input a food tray image without the label to teach the system. Then we input a food tray image with the label and expect that the system can automatically detect the position and the orientation of the label based on the information it is taught.

Our method contains three major steps:

1. Feature extraction using the Speeded Up Robust Features (SURF) algorithm Bay *et al.* [2006];
2. Feature matching and image registration using the Random Sample Consensus method (RANSAC) Fischler & Bolles [1981] and the Levenberg-Mardquardt algorithm;
3. Label region localisation and repairing.

In the following, we describe each step in details.

3.3.1 Feature extraction via SURF

The proposed method requires two input images. As shown in Fig. 3.1, one is a base image of a food tray without the label and the other is a test image of a

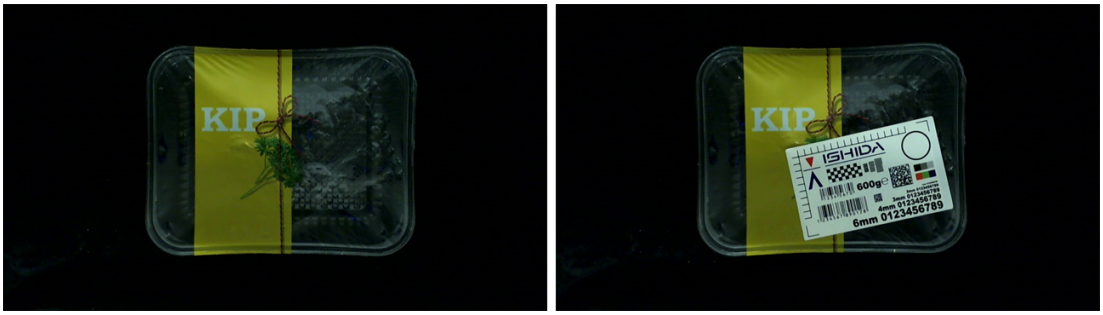


Figure 3.1: The two input images required by our method

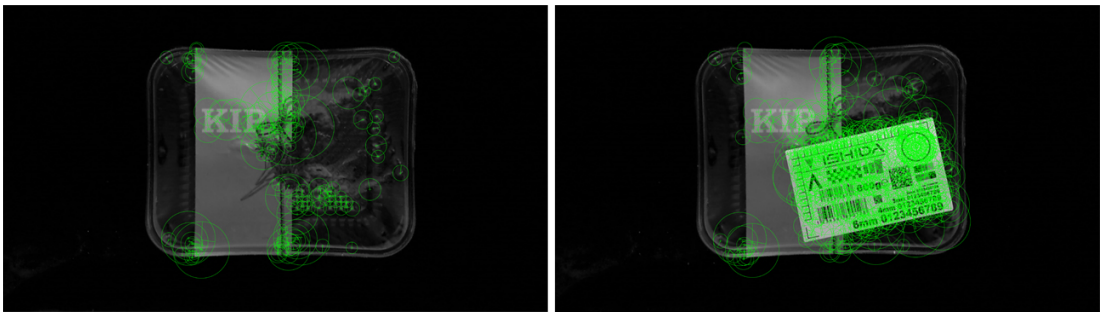


Figure 3.2: The results of feature extraction using SURF where the centre of each circle denotes the location of each feature, the size of the circle denotes the scale of the feature and the line segment within the circle denotes the orientation of the feature.

food tray with the label. Then, we employ the SURF method to perform feature extraction. Please refer to our report of November 2013 for the details of the implementation of the SURF method. Here, we just show the results of our implementation in Fig. 3.2.

3.3.2 Feature matching and image registration

Fig. 3.3 shows the result of the raw feature matching produced by SURF where we use the yellow lines to illustrate the matches. It can be seen that some features points are not correctly matched. A correct match means that the two matched points should correspond to the same point in the real world although they are from different images. Incorrect matches can lead to a poor registration between

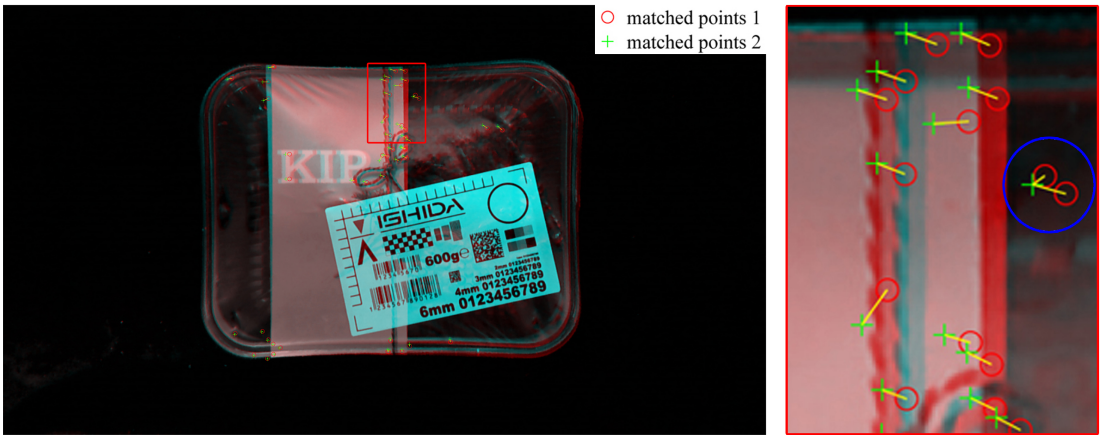


Figure 3.3: The raw matches produced by SURF contains some incorrect matches. To visualise the results, we deliberately colour the two images in red and blue respectively. In the blue circle marked in the enlarged subfigure on the right hand side, there are incorrect matches since two points cannot correspond to the same point.

the two images. To address this problem, we employ the RANSAC method to remove incorrect matches. The details of the RANSAC method have been given in our report in November. The outcome of applying it is a refinement of the matches, only preserving the correct matches as shown in Fig. 3.4.

The image registration via Levenberg-Mardquardt algorithm has also been mentioned in our last report. Basically, we utilise this technique to align the base image and the test image in a uniform coordinate system. It is worth noting that in this specific task, the registration is actually dependent on the matched feature points in the non-label regions since they are the common regions contained by both images. Thus one assumption of the method is that the non-label regions of the tray contain enough number of features. In practice, such an assumption can hold because (i) the films of food trays usually contain some printed patterns or (ii) there is some food in the tray which actually forms some visual patterns even if the film is completely transparent.

The outcome of the registration is shown in Fig. 3.5. It can be seen that each pair of corresponding feature points have already been moved to the same position after the registration. This offers us a good basis for the next step.

Figure 3.4: By using RANSAC, we remove incorrect raw matches or 'outliers', leaving only the 'inliers'.

3.3.3 Label localisation and repairing

To localise the label region, we first normalise the intensity values of the registered base image and the test image for the later image subtraction. To make the subtraction not very sensitive to the ever-present image noise (mostly caused by the sensor), we apply an average filtering to the subtraction image where the size of the convolution kernel is set to 10×10. This leads to Fig. 3.6 where the subtraction image looks a little bit blurred because of the effect of filtering. To achieve a better visualisation, we also normalise the subtraction image to interval $[0, 1]$. Thus in Fig. 3.6, the warmer the colour, the larger the difference between the base image and the test image.

To better localise the label, an intuitive idea inspired by Fig. 3.6 is to set a threshold and then detect the pixels with values greater than the threshold as the label region. Unfortunately, this idea is rather problematic. This is because the values are not consistent within the label region. As shown in Fig. 3.6, the top right corner has a strong response while quite a few regions inside the

Figure 3.5: After registration, the feature points from different images are fully overlapped.

label are in blue. A thresholding scheme is thus not robust enough to detect a complete rectangular region as the label region. Instead, we develop a method to more reliably detect the repaired rectangular lable region using image erosion and dilation.

First, we convert the subtraction image to a binary image by thresholding. The threshold is set as the double of the mean value of the subtraction image. Second, a flood-fill operation is performed on the binary image to fill the holes. A hole is a set of background pixels which cannot be reached by filling in the background from the edge of the image. We then perform image erosion by creating a disk-shaped morphological structuring element. The radius of the disk is fixed to 70. Next, we implement image dilation using the same morphological structuring element. To pursue an adaptive scheme, we iteratively adjust the threshold used for the binarisation of the subtraction image. In each iteration, we measure the number of the connected components in the image after the operations of erosion and dilation. If it is not equal to 1, the threshold is added by 0.1 and the aforementioned steps will be repeated. Such an adaptive scheme

Figure 3.6: The result of image subtraction using the registered base image and the original test image.

guarantees that one complete label region can be detected. The final step is to draw the minimum bounding box around the detected label region by computing its convex hull. Fig. 3.7 shows the final result of our label detection method.

3.4 More results

Figs. 3.8–3.12 show more results where we can see that the proposed method is highly reliable and the positions and the orientations of the trays are various. In general, the running time is 1-2 seconds using an Matlab implementation. We expect that the speed could be much faster if we can develop a C++ implementation. Certainly, before doing so, more tests need to be done to demonstrate that the current version of this method is really reliable and to make sure that the method is flawless.

Figure 3.7: The final result of label region detection

3.5 Future work

Besides the redevelopment of the proposed method in C++ in the near future, more functionalities can be added. For example, we can teach the system what is the right position of the label by inputting a food tray image with a label at the right place. Then, by combining our previous development on background segmentation, the system should be able to recognise whether the label position is right for any input food tray images. Another issue is to obtain more datasets to test the reliability of the system. In particular, we need to establish our own ground truth data to evaluate the accuracy of the system.

Figure 3.8: Top left: the base image; Top right: the test image; Bottom: label detection

Figure 3.9: Top left: the base image; Top right: the test image; Bottom: label detection

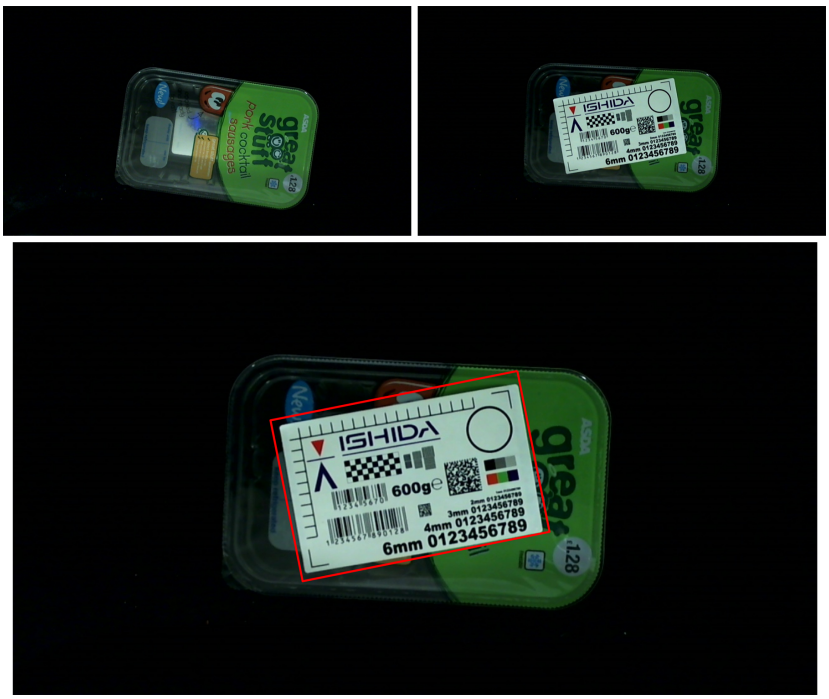


Figure 3.10: Top left: the base image; Top right: the test image; Bottom: label detection

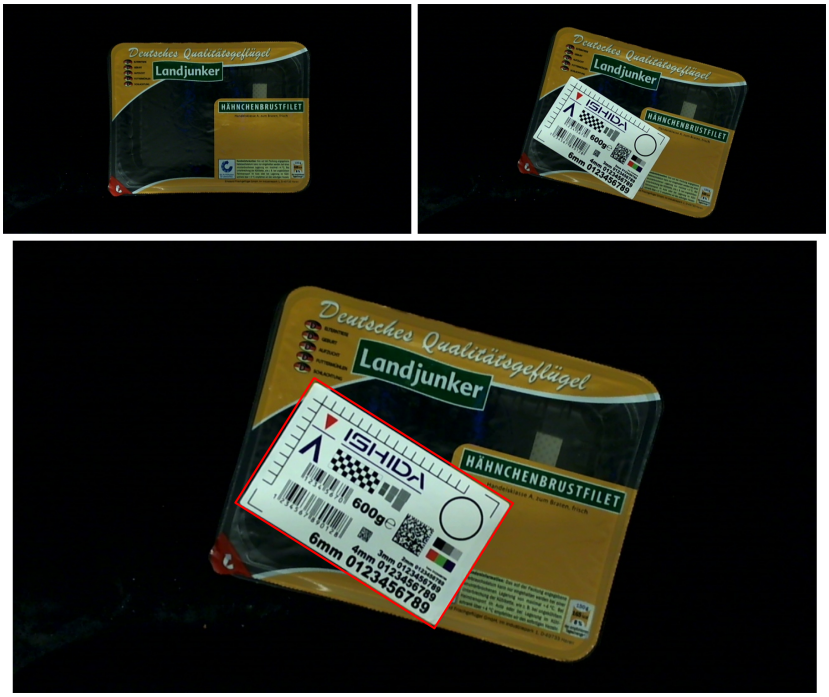


Figure 3.11: Top left: the base image; Top right: the test image; Bottom: label detection

Figure 3.12: Top left: the base image; Top right: the test image; Bottom: label detection

Chapter 4

Ran Song TADD Technical Report for Jan 2014

4.1 Abstract

In December 2013, we reported an automatic method in order to detect the region of a label within a tray image, given that that the tray image without the label has been provided. In this report, we extend the method to solve the following problem: how can we automatically and robustly evaluate the position and the orientation errors of a label within any tray image? Preliminary experiments show that the proposed method is efficient, reliable and easy-to-use (in particular for non-experts) while certainly more tests need to be done to comprehensively evaluate it.

4.2 Introduction

In this report, we describe our newly developed method for automatically detecting the position error and the orientation error of a label within a test image of a food tray, given that two reference images have been provided. The first reference image is a tray image without any label and the second one is a tray image with the label at the right position. The basic idea is to teach the system by first inputting the two reference images and then using the learned knowledge

to compute the errors of the test image with regard to the reference images.

4.3 Method

The method contains three major steps. First, we register the second reference image to the first reference image and detect the rectangular label region using the technique that we proposed in our last monthly report. Second, we employ the same technique to register the test image to the first reference image and also detect another rectangular label region. The third step is to measure the positional difference between the two rectangular regions, defined by the so-called 'position error' and 'orientation error'.

We here just simply review the technique for registration and label region detection. It includes three components:

1. Feature extraction using the Speeded Up Robust Features (SURF) algorithm Bay *et al.* [2006];

2. Feature matching and image registration using the Random Sample Consensus method (RANSAC) Fischler & Bolles [1981] and the Levenberg-Mardquardt algorithm;

3. Label region localisation and repairing.

Note that the method proposed in our last report is designed to deal with the images captured by a hand-held camera where their backgrounds are black. There are some changes in the latest version of our software to make sure that it can handle the images (for example, we show the three input images in Figure 4.1) captured by the camera mounted on the real Ishida machine in Witham Wharf in Lincoln.

1. Set the intensity values of the pixels around the border region of the registered images to $I(500, 1) - 10$ where $I(500, 1)$ represents the intensity value of the pixel with row number=500 and column number=1 in the first reference image. In the previous version, the values are set to 0 by default.

Figure 4.1: The three input images. Top: the first reference image; Middle: the second reference image; Bottom: the test image

Figure 4.2: The inconsistency within the background region (in particular, the border regions pointed by the red arrows) is compensated by resetting the pixels values.

2. Give a stronger blurring to the difference image computed by subtracting the registered images. In the previous version, we blur every pixel with its neighbours in a 10×10 neighbourhood while in the latest version, we use a 40×40 one.

3. Adjust the threshold by 25% which converts the difference image from an intensity image to a binary image.

With the help of these changes, we solve the problem caused by the inconsistent background illustrated in Figure 4.2. And this leads to the successful detection shown in Figure 4.3. In Figure 4.3, the yellow rectangle denotes the position of the label in the test image and the blue one denotes where it should be (the correct position). Our software also outputs a list of statistics as shown in Figure 4.4, which can be used to quantitatively evaluate whether the position of the label in the test image is wrong and how wrong it is.

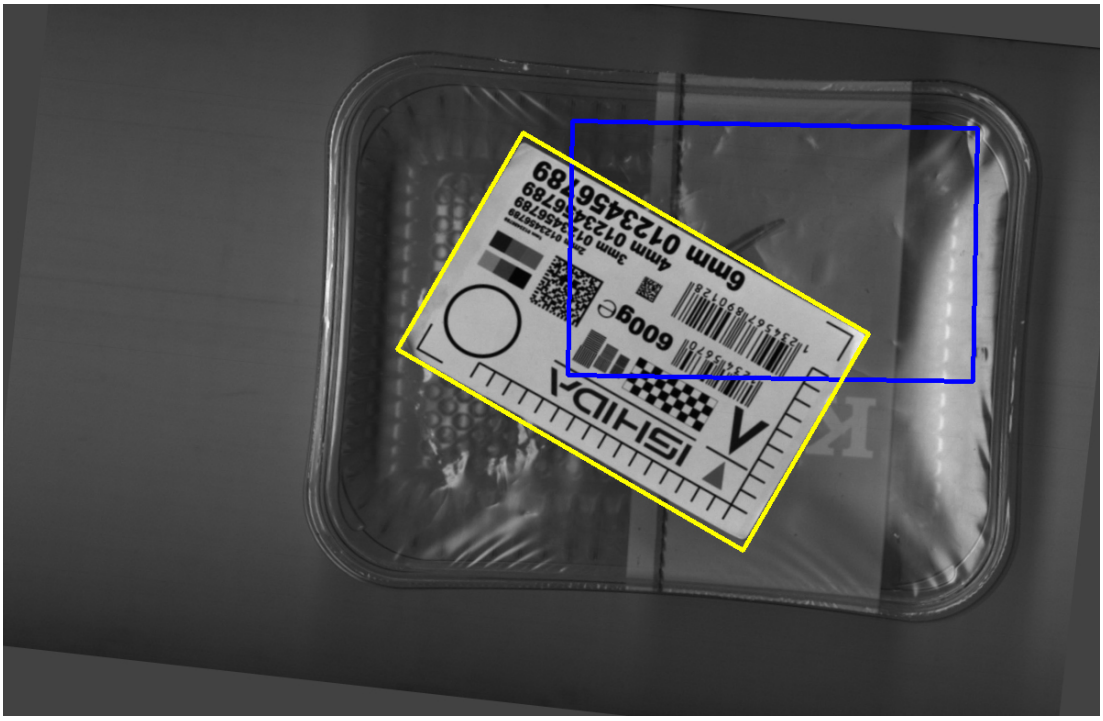


Figure 4.3: Visualisation of label region detection.

4.4 More results

More implementation results are shown in Figures 4.5 and 4.6 where the top rows show the input images and the bottom rows show the detection results. In particular, in Figure 4.5, we input a test image which is exactly the same with the second reference image. As we expect, the position and the orientation errors are both equal to 0 and the two rectangles are overlapped with each other.

4.5 Future work

One limitation for the current software is that setting the pixel value to a specific one, such as $I(500, 1) - 10$ that we used in this work, is not reliable. There would be a larger possibility for the software to fail if the rotation angel is too large since it would produce larger inconsistent border regions (please refer to Figure 4.2 for a better understanding). Thus currently, we assume that all rotation angles

```
C:\Users\computing\Documents\Visual Studio 2010\Projects\imagerega\Debug>imagere
g img0000.jpg img0001.jpg img0002.jpg
Centre point: [839.396, 269.404]
Orientation: -89.0317
Centre point: [686.884, 367.722]
Orientation: -59.8935
Position error: 181.457
Orientation error: 29.1382
```

Figure 4.4: The statistics of label detection. The 'Centre point' displays the coordinates of the centre point of the label and the 'Orientation' displays its orientation angel. The 'Position error' is calculated by the distance between the two centre points and the 'Orientation error' is measured by the included angle between the two orientation angles.

Figure 4.5: The results of label detection

are smaller than 45 degree although it is highly likely to be so in practice. A reliable solution is to change the conveyor belt of the Ishida machine to a black one. Secondly, the current software requires manual input. The future work will try to get it connected with the Ishida machine with running conveyor and automatically read the input images captured by the camera mounted on the machine. This seems not trivial. Simply checking the time stamp of the image saved on the disk does not work since its precision is just minutes (it could be seconds if we use some external library but it is still not enough). Within one

Figure 4.6: The results of label detection

minute or one second, the image captured by the camera and then saved on the disk could be modified since the camera has already taken another shot. Also, to use such a strategy, the software needs to frequently check the time stamp of the image, it will slow down the processing. A reliable solution is to look up the software attached to the camera to check whether a certain flag (a parameter in BOOL type) is changed (e.g., from 0 to 1) whenever it takes a shot. It is known that for a webcam, such a flag does exist and it can be called within the framework of OpenCV.

Chapter 5

Hossein Malekmohamadi TADD Technical Report for Nov 2013

5.1 Introduction

This is the November report for the project Trainable Vision-based Anomaly Detection and Diagnosis (TADD). The aim of this project is to deliver a software system capable of detecting and recognising intelligently different anomalies in food contents and packages, e.g. potato or food container. Currently, this system can do semantic segmentation, barcode region detection and foreground extraction based on some image processing techniques and the Adaboost learning. These features can be used in different parts of quality control task which is an industrial aim in TADD. However, further improvements to TADD can be accomplished by bringing 3D vision systems and enhancing learning capabilities. In this report, the focus is on candidate techniques in machine vision and machine learning for TADD systems. This is covered in Section 2 where we investigate photometric stereo as the candidate 3D vision system and random forests and deep learning as the candidate learning algorithms. Finally, Section 3 is a summary of this report.

5.2 Candidate Techniques for TADD

Here, an overall review is given for further improvements in TADD systems both in machine vision techniques and learning algorithms. For the vision part, there are several possibilities from expensive solutions to cost-effective ones. We had visitors from Cognex to show us their scattering laser imaging system. We plan to visit Sick UK to see their demo in the near future. However, another candidate to obtain more details of potato and food packaging is to build a photometric stereo rig. Photometric stereo can be built cost effectively and it will reveal 3D dense information of surfaces under investigation. Here, we give a brief overview on photometric stereo algorithm and some initial results for the data gathered in Kingston university. Furthermore, improvements to TADD system can be accomplished in the machine learning side of the project with topics like decision forests or deep learning. In this chapter, a brief introduction to random forests and deep learning algorithms are given.

5.3 Photometric Stereo

To recover highly dense surface information, we employ the Photometric Stereo (PS) technique which is based on high-resolution image acquisition using a number of different light source directions and a single 2D camera as shown in Figure 5.1. PS is used to estimate the dense normal map of a potato or a package, from which the gradient field is then computed. This enables robust separation of the 2D (albedo) and 3D (height map) components of the potato or package at a significantly high level of accuracy.

PS has been available for many years but only recently has affordable technology become available for improving camera resolution and algorithm execution speed, allowing synchronised light switching at the fast rates needed to avoid inter-frame motion. We use 4 photometric images (4 source PS system) to recover the 3D surface information from potato. While fewer photometric images could be sufficient, a 4 source PS system allows better recovery in the presence of highlights and shadows . We assume a Lambertian reflectance model together

with intensity variation at each pixel to estimate local surface orientations. The integration of these surface orientations results in a highly detailed estimate of the surface geometry. The captured images per each LED illumination are shown in Figure 5.2.

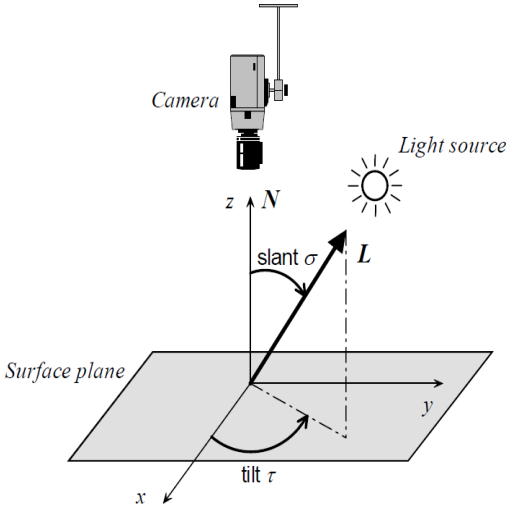


Figure 5.1: Typical PS Set-up and the Surface Under Investigation

Figure 5.2: 4 Captured Images

Figure 5.1 shows the set-up for the capture of photometric images. τ denotes tilt angle and represents the angle that an illuminant vector projected onto the

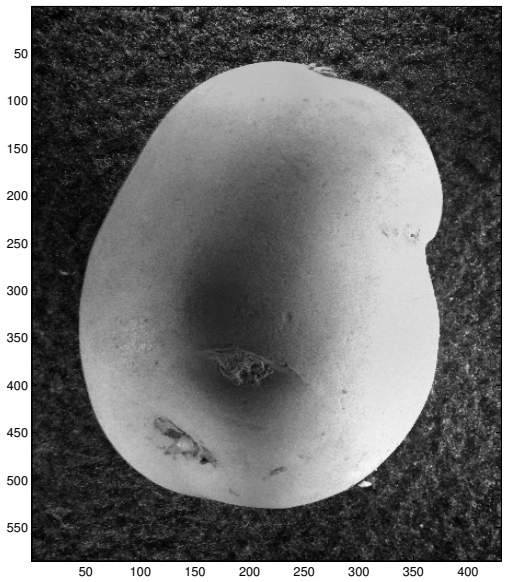


Figure 5.3: Albedo Image from Kingston Data (Code is implemented in MATLAB)

surface plane makes with *x-axis*. The slant angle, σ, represents the angle the illuminant vector makes with the *z-axis*. The function to represent the photometric images at individual illumination direction is as follows:

$$i(x,y) = \frac{-p(x,y)cos\tau sin\delta - q(x,y)sin\tau cos\delta + cos\delta}{\sqrt{p(x,y)^2 + q(x,y)^2 + 1}} \tag{5.1}$$

It is important that the three photometric images provide enough change in illumination gradient so that the partial derivatives for the surface (p and q) can be estimated Woodham [1980]. Based on the assumption that Lambert's law is preserved, we model these reflectance functions for the 4 PS images as

$$i_d(x,y) = \rho(l_d n) \qquad \forall d \in \{1,2,3,4\} \tag{5.2}$$

With known reflectance intensities i_d and illumination directions l_d (which are fixed in the camera coordinate), the unit surface normal at a given position (x,y) in the surface plane is given by $n = \frac{(p,q,-1)^T}{\sqrt{p^2+q^2+1}}$ with p and q being the partial surface

derivatives. Having surface normal components, it is time to reconstruct the surface as shown in Figure 5.4. Among different integration algorithms to generates integrable surface from gradients, we apply Frankot-Chellappa methodFrankot & Chellappa [1988].

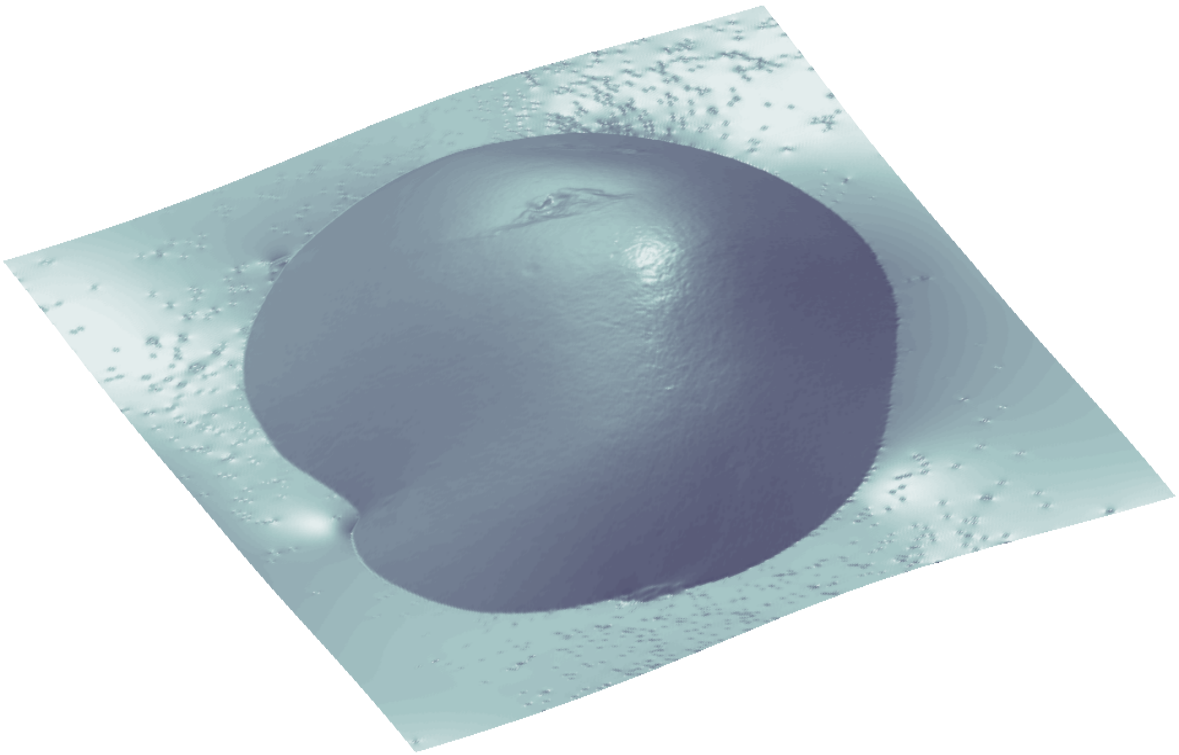


Figure 5.4: Reconstructed Surface Height Map (Code is implemented in MATLAB)

Following the extraction of surface gradients, we need to derive some features to be fed in to machine learning algorithm. One candidate for the feature extraction stage is to use the partial derivatives of the 3D surfaces (p,q) to extract surface curvature information. We then compute the first order derivatives along the *x-axis* and *y-axis* for p and q: p_x, q_y, p_y. These derivatives are used to compute the surface curvature. The Gaussian curvature, K, is given by

$$K = \frac{p_x q_y - {p_y}^2}{(1 + p^2 + q^2)^2} \tag{5.3}$$

Mean curvature is

$$H = \frac{(1+p^2)q_y + (1+q^2)p_x - 2pqp_y}{2(1+p^2+q^2)^{\frac{3}{2}}} \tag{5.4}$$

The principle curvatures $k_{1,2}$ are

$$k_{1,2} = H \pm \sqrt{H^2 - K} \tag{5.5}$$

The resultant principal curvatures as shown in Figures 5.5 and 5.6 are used to compute a shape index (SI) map for each 3D surface. This provides a characterisation of topography using a continuous angular representation. Shape index is defined as

$$S = \frac{2}{\pi} arctan(\frac{k_2 + k_1}{k_2 - k_1}) \tag{5.6}$$

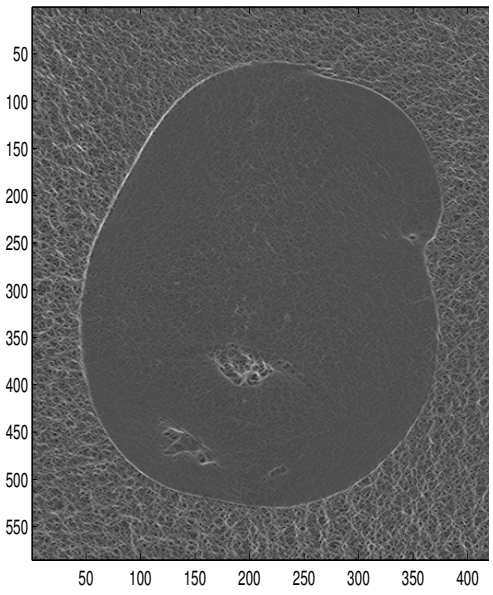


Figure 5.5: Principle Curvature 1 (Code is implemented in MATLAB)

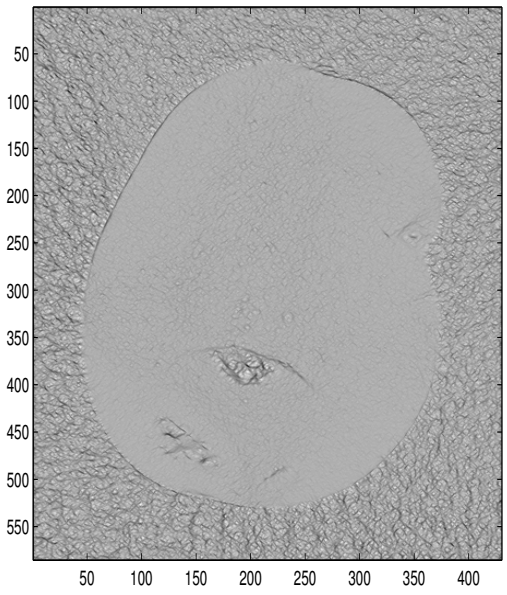


Figure 5.6: Principle Curvature 2 (Code is implemented in MATLAB)

In Figure 5.7 we illustrate the shape index maps of a potato from the Kingston

dataset[1]. The shape index maps presented appear to be able to capture significant amount of 3D surface texture information. From the shape index map, we can clearly see that the texture information of blemish area.

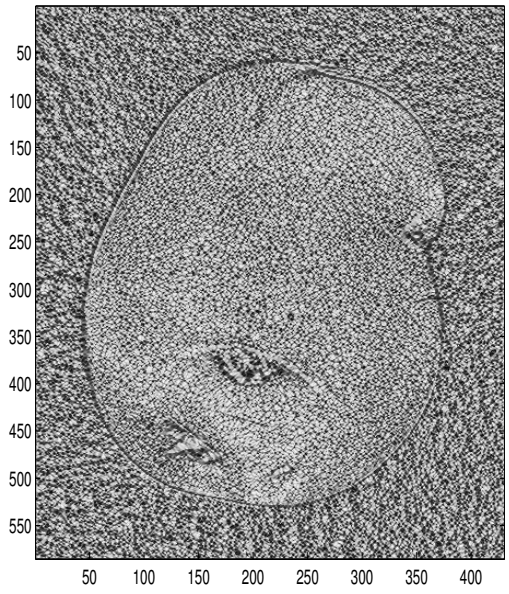


Figure 5.7: Shape Index Map (Code is implemented in MATLAB)

We have implemented the PS technique and 3D surface information in MATLAB. In this implementation, the inputs are 4 images as shown in Figure 5.2 and the outputs are surface normals, height map, principle curvatures and shape index. Briefly, the PS technique can be applied in our current TADD system to model 3D objects. Its applications are various from potato disease detection to seal integrity identification. However, the choice for white LEDs as point source lights and a high resolution camera should be considered within the next quarter.

[1]Images are captured by Dr Vasileios Argyriou at Kingston University in November 2013

5.3.1 Potential Improvements to the Learning Section of TADD

Current system benefits from Adaboost classification algorithm however we are looking for increasing speed and accuracy. In this part, we give a brief overview for two learning candidates for this project: 1) Decision forest and 2) Deep learning. Both techniques have absorbed a lot of attention in the research community in the last decade. Decision forests have light-weight implementations and operate fast. They are suitable for tasks like segmentation. Deep learning is to able learn multiple levels of representation and abstraction that help to make sense of data like images of sealed trays with food content.

- Decision Forests are amongst estimative methods that can classify data. Decision forests consists of several decision trees (DTs). A decision tree uses training data to construct a classifier, which models the relations between data attributes and class targets. The learning process of decision trees is an inductive process that uses particular facts from data attributes to make more generalised conclusions Witten *et al.* [2011]. Once it has learnt, the decision tree is able to estimate the output based on new inputs. The final model is made of a set of Boolean tests, which is less complex than a one-stage classifier for implementations and real-time tasks. Representing a decision tree can be performed two ways. In the first method, a decision tree is modelled with the influence diagram (ID) through nodes and arcs. There are three types of nodes: 1) decision nodes represent points at which the system has to make a choice of one alternative from a number of possible alternatives; 2) chance nodes represent points at which chance, or probability, plays a dominant role and reflect alternatives over which the system has no control; and 3) terminal nodes represent the ends of paths from top-down through the decision tree. Both decision and chance nodes are interior nodes and can have child nodes, but terminal nodes have no child nodes. Arcs are in three types: functional which ends in a decision node, conditional and informational arcs. Second, mathematically, decision trees are If- Then rules, where different rules apply sequentially to get a specific output.

There are different algorithms for the learning process of a decision tree. Common to all the algorithms is that a decision tree chooses the best attributes to split the instances based on some measures like information gain or gini index Breiman [1993]. If the stopping criterion is met, the splitting attribute represents a decision node. The stopping criterion is when there is no attribute or training sample left or all samples have the same target value. Due to numerous attributes and targets, handling error in training a decision tree is performed mainly by pruning; e.g. if two decision trees employ the same attributes and have the same accuracy, the one with fewer leaves (nodes) is selected for the final model. Pruning can be performed either as pre-prune (forward pruning) or as post-prune (backward pruning). In pre-pruning, it is achievable to stop adding new attributes and hence hinder over sizing through some measures like information gain. Post-pruning is a recursive task that it waits for the full decision tree has built and then prunes the attributes by two major concepts: 1) sub-tree replacement; and 2) sub-tree raising. Post-pruning is an inherent feature of many decision tree learning algorithms to obtain optimum tree with an acceptable rate of accuracy.

There are different methods for learning process of decision trees like C4.5, ID3, and CHAID Witten *et al.* [2011]. In this research C4.5 is selected. Based on some information theory concepts like entropy, C4.5 builds decision trees from a set of training samples considering error rate and speed. C4.5 is a greedy, recursive and top-down algorithm for decision trees and has an appropriate combination of speed and accuracy. C4.5 starts with the root node where it selects an attribute more capable of separating the instances. Following the root node, C4.5 decides on other attributes recursively until it meets the stopping criterion and where all training instances are classified. The selection measure is based on information theory and that is information gain. Lastly, post-pruning is accomplished to obtain the optimum tree with fewer leaves and higher accuracy.

Random forests is another machine learning techniques which consists of several weaker classifiers (decision trees) Breiman [2001]. Weaker classifiers can be combined by voting (for classification) or averaging (for regression) to form strong

classifier. It is applicable for classification, regression, clustering, density estimation, outlier and anomaly detection. A random forest has some specific rules for tree growing and combination, self-testing and post-processing Breiman [2001]. Consequently, they can deliver higher prediction accuracy for even very large volume of data and unlike decision trees they are resistant to the over-fitting problem.

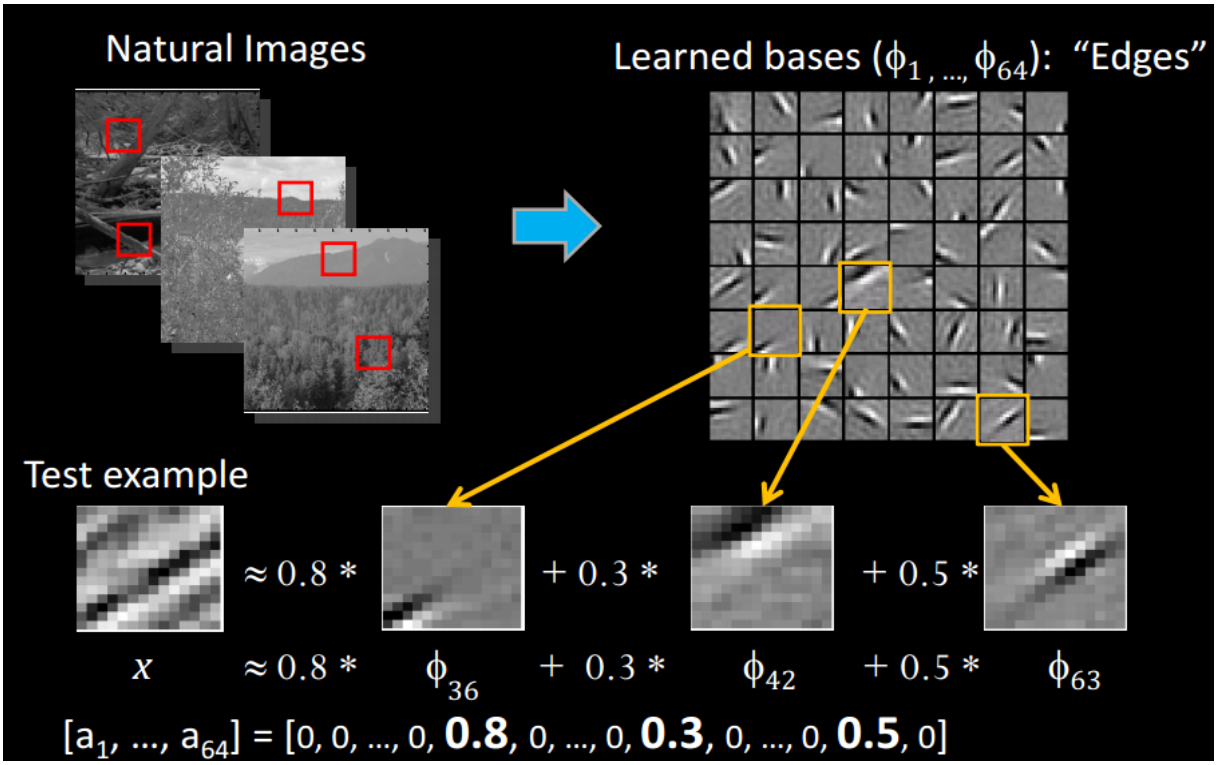


Figure 5.8: Sparse Coding Illustration (Image is taken from Ng [2013])

-Deep Learning is a set of machine learning algorithms that attempt to learn in multiple levels, e.g. neural network layers, corresponding to different levels of abstraction Ng [2013]. In multi-level learning, higher-level concepts are defined from lower-level ones, and the same lower-level concepts can help to define many higher-level concepts. In deep learning, an instance, e.g. an image, can be represented in several ways depending on the extracted features, but some representations make it easier to learn tasks of interest. Features extracted based on SIFT, Spin image, HoG, RIFT, Texton and GLOH are widely applied to represent

vision features. For the first stage in deep learning which is feature extraction, each image is represented by a dictionary of features, e.g. edge detection in a patch of image, using sparse coding. As shown in Figure 5.8, images can be represented in higher level than the raw pixels. In the next higher levels, combination of edges are mixed to represent object parts, e.g. eye. This hierarchical set-up can be implemented with neural networks to classify image contents, e.g. car, chair, plane, etc.

In the TADD system, depending of application layer, we can implement any of the above-mentioned algorithms to increase accuracy, speed and robustness. For example, deep learning can be helpful in semantic segmentation or random forests can be applied in blemish detection.

5.4 Summary and Future Work

In this report, we investigated several potential techniques to be implemented in the TADD systems. These techniques are either for image acquisition and set-up like adding 3D vision or enhancing the learning side of the software package. For the vision systems, there are some expensive sensors which can be bought within project budget like the sensor we had been shown by Cognex. These sensors recovers surface and then based on that we can estimate surface normals. However in some cases, we need to investigate 3D surface normals first and then, if necessary recover the height map. This can be achieved by basic mathematical model called photometric stereo. In this method, surface normals are extracted based on a set-up containing at least three LED light sources and a high definition 2D camera. For the learning part of TADD, we can enhance the speed and accuracy by other algorithms. In this report, deep learning and random forests are presented. Deep learning can be more suitable for tasks like barcode region detection or date detection in different languages. Decision trees can be more useful in potato blemish detection where speed is more important.

The aim in the next quarter of this project is to investigate the aforementioned techniques. For this purpose, initial studies and research on Kingston University

data have been done and we are investigating the building of our own PS rig for 3D surface reconstruction of potato and food containers. Moreover, manipulating the current codes in TADD, we can have access to data before current learning algorithm (Adaboost) and then we can test random forest learning.

Chapter 6

Hossein Malekmohamadi TADD Technical Report for Dec 2013

6.1 Introduction

This is the December 2013 report for the project Trainable Vision-based Anomaly Detection and Diagnosis (TADD). The aim of this project is to deliver a software system capable of detecting and recognising intelligently different anomalies in food contents and packages, e.g. potato or food container. Currently, this system can do semantic segmentation, barcode region detection and foreground extraction based on some image processing techniques and the Adaboost learning. These features can be used in different parts of QC which is an industrial aim in TADD. However, further improvements to TADD can be accomplished by reinforcing semantic segmentation, applying 3D/UV imaging apparatus and enhancing learning capabilities. In the November report, we gave a brief introduction to a possible 3D vision system and decision forests. In this report (December), the focus is on candidate technique for reinforcing semantic segmentation for different data sets.

6.2 Superpixel Segmentation

To segment high resolution images of potatoes, we employ superpixel segmentation Achanta *et al.* [2012]. Here, we applied this technique to various datasets to create images for training and testing the classifier. The aim of the classifier can be identifying blemish in potatoes or label detection in trays. For this purpose, we need some ground truth images which are available for datasets 1 and 2 in the list below:

- Potatoes (Example is show in Figure 6.1): This data set is recorded in a PS set-up (please refer to November 2013 report). Each potato has 4 digital images corresponding to 4 LEDs.
- White potatoes (Example is show in Figure 6.2): Each white potato has a marked image, where experts have been asked to colour code different regions corresponding different blemishes.
- Red potatoes (Example is show in Figure 6.3): Like white potato dataset, each red potato has a marked version.
- Trays (Example is show in Figure 6.4): Different sealed trays are available with or without labels.
- Grapes (Example is show in Figure 6.5): Different grapes and stalks have been captured with DSLR to investigate texture.
- Pittsburgh fast-food image dataset (PFID) Chen *et al.* [2009] (Example is show in Figure 6.6): This dataset contains restaurant (low-res) and lab (hi-res) images for different fast food chain stores. It focuses on different textures for food contents.

For each dataset, we used superpixel segmentation Achanta *et al.* [2012]. For white and red potato datasets, segmentations are applied to their unmarked images. Examples are shown in Figure 6.7-Figure 6.10. Afterwards, we use ground truth images, if available, for potato blemish detection or food texture analysis. Example in separating marked images to create different mask images is shown in

Figure 6.11. We can define global or local features before learning algorithm. As an example, image contrast analysis to compare healthy and blemish regions is a suitable candidate. On the other hand, having 3D information from Kingston data (or in future from other 3D setups) and superpixel segmentation, we can classify different regions in food contents for different purposes. This is another goal for the first quarter of 2014.

Figure 6.1: An image of potato under different illuminations recored in Kingston university as appeared in November 2013 report.

We have implemented the codes in this report in MATLAB. In addition, we have used VLfeat library published in Vedaldi & Fulkerson [2010] for superpixel segmentation.

6.3 Conclusions

In this report, we applied superpixel segmentation technique to various datasets. We use this information together with statistical or structural image features to train classifier to detect potato blemishes or label detection. The data collected from a 3D setup can also be used with segmentation to a robust classification algorithm for either blemish detection or food texture analysis. As an example, healthy parts of potato have less standard deviation for tilt angles in 3D reconstruction compared to unhealthy regions. The aim in the next quarter of

Figure 6.2: An image of a white potato and expert-marked version of it.

this project is feature extraction both in 2D and 3D. Extracted features can be applied locally or globally. An example of local feature is to measure local auto-covariance in blemish areas to be fed into a supervised learning. Prior to applying 3D features to TADD, we need to have a base version of TADD that works without dependencies on GPU and specific machine. Afterwards, we will be investigating using different 3D sensors and their API's in C++.

Figure 6.3: An image of a red potato and expert-marked version of it.

Figure 6.4: Image of two different trays recored in the current TADD machine in UoL.

Figure 6.5: Images of different grapes.

Figure 6.6: Sample images of PFID dataset.

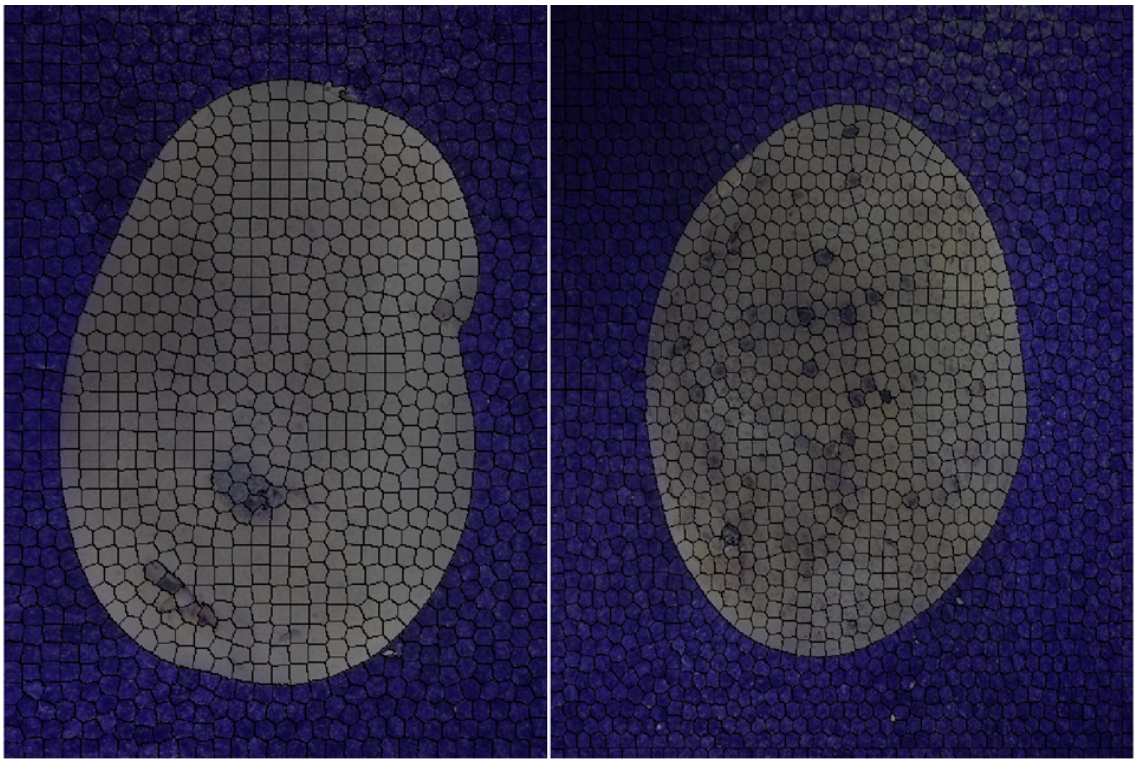


Figure 6.7: Sample images from Kingston dataset (average image) where overlaid superpixel segmentations are also shown.

Figure 6.8: An image of a white potato and expert-marked version of it where overlaid superpixel segmentation is also shown. Superpixel segmentation is done for unmarked and the map is then overlaid to marked image.

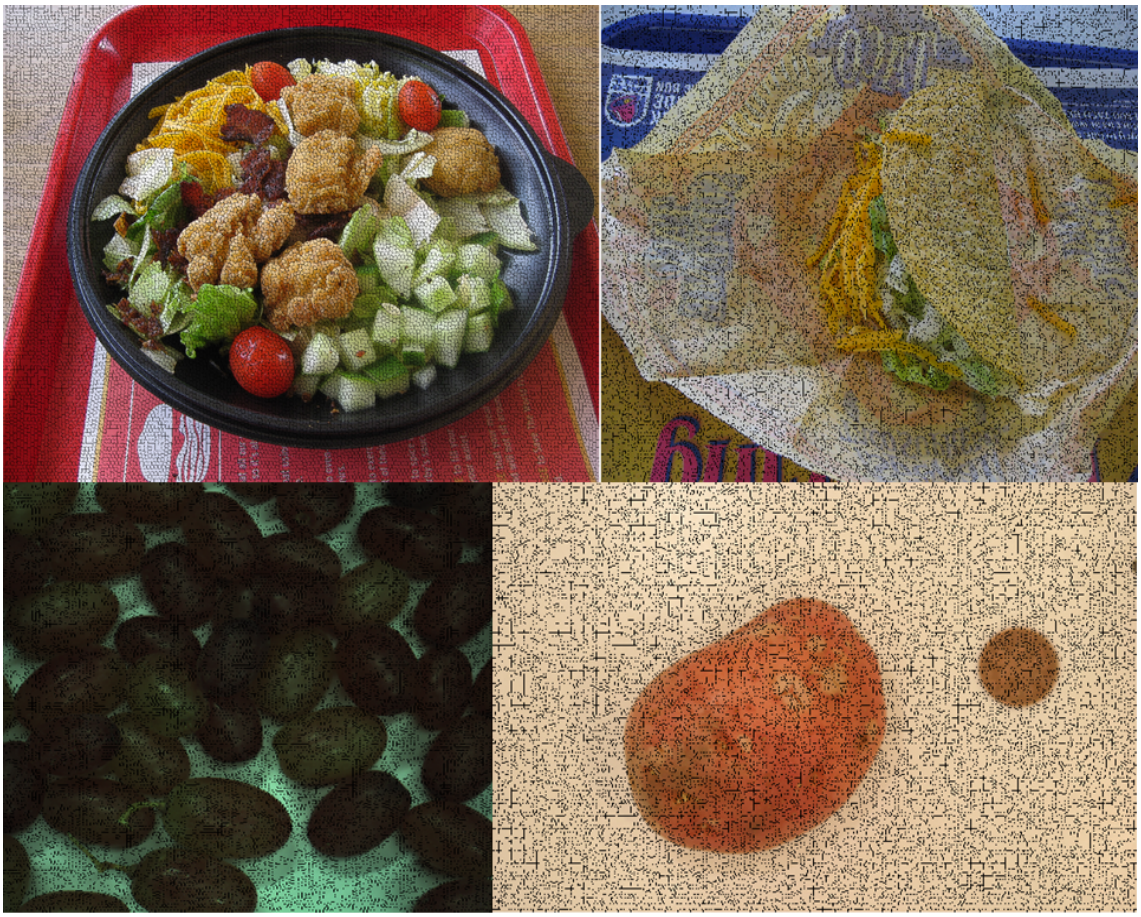


Figure 6.9: Sample images from PFID dataset where overlaid superpixel segmentations are also shown.

Figure 6.10: Sample images from PFID dataset where overlaid superpixel segmentations are also shown.

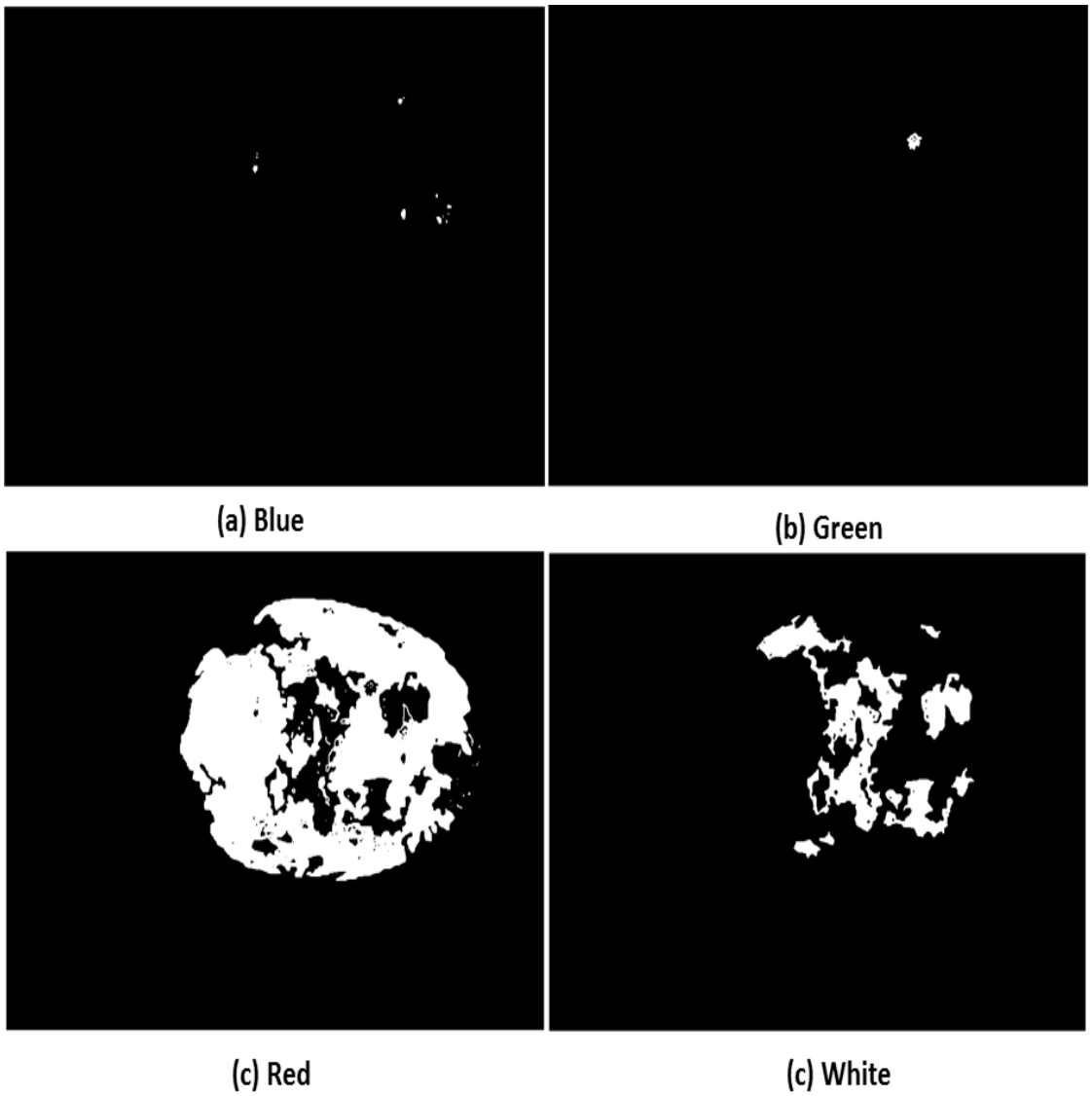
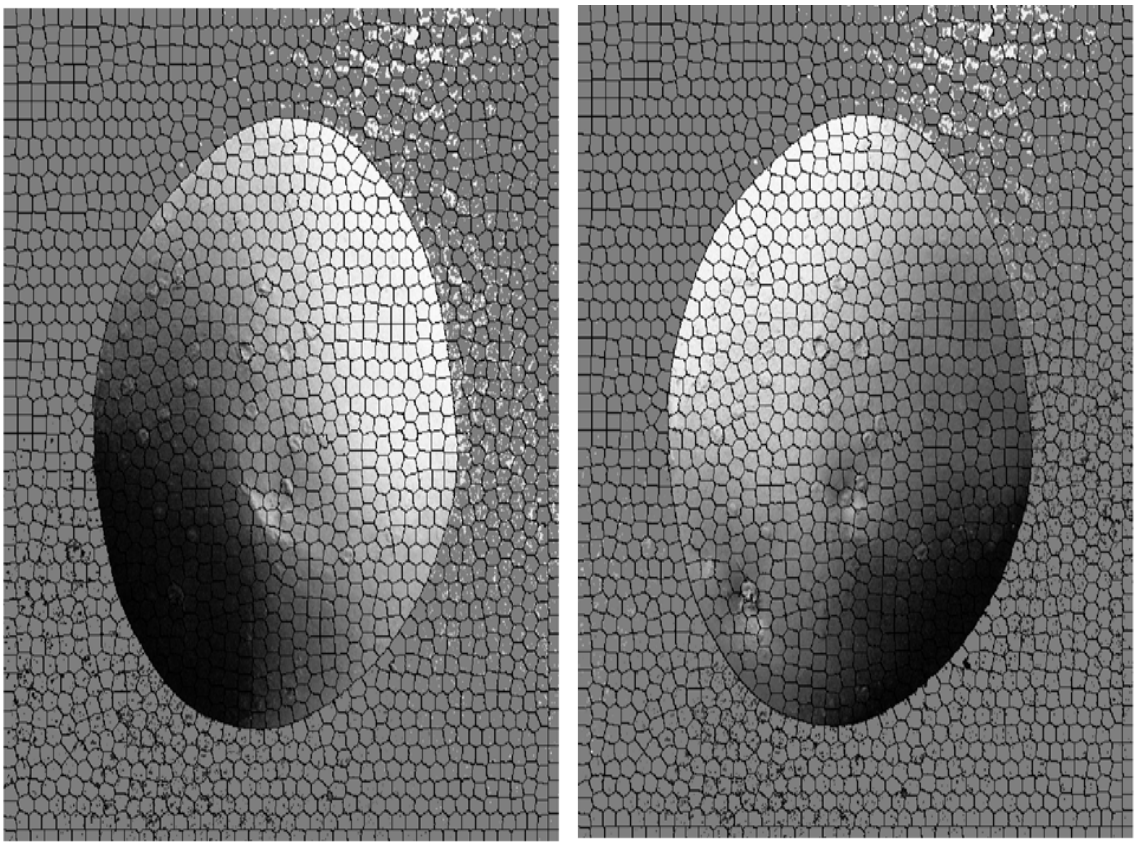


Figure 6.11: Mask images of a candidate marked white potato are extracted to correspond different blemishes. We extract structural and statistical image features globally (whole image) and/or for each individual marked region to use it in a classifier.



(a) P: Surface derivative along x axis

(b) Q: Surface derivative along y axis

Figure 6.12: Surface derivatives of a potato from Kingston University data are mapped with segmentation data extracted from albedo image. We extract 3D texture information per region (superpixel) to distinguish between blemish and healthy regions as well as food content analysis, e.g. bump map for potato is different to a cauliflower.

Chapter 7

Hossein Malekmohamadi TADD Technical Report for Jan 2014

7.1 Introduction

This is the January 2014 report for the project Trainable Vision-based Anomaly Detection and Diagnosis (TADD). The aim of this project is to deliver a software system capable of detecting and recognising intelligently different anomalies in food contents and packages, e.g. potato or food container. Currently, this system can do semantic segmentation, bar-code region detection and foreground extraction based on some image processing techniques and the Adaboost learning. These features can be used in different parts of QC which is an industrial aim in TADD. However, further improvements to TADD can be accomplished by reinforcing semantic segmentation, applying 3D/UV imaging apparatus and enhancing learning capabilities. In the November report, we gave a brief introduction to a possible 3D vision system and decision forests. In the December report, we created training and test datasets for semantic segmentation for further research based on multiple datasets. In this report (January), the aims are investigating 3D recording devices and extracting 3D features to analyse food contents. Prior to this, we have to modify TADD codes to be executable on any machine that requires cutting dependencies on GPU processing for the superpixel segmentation.

7.2 TADD Software

The current TADD hardware includes a low-cost 2D vision sensor and a standard desktop computer with a graphics processing unit (GPU). The software algorithms mainly based on prior research reported in Hutton [2012] to enable detection, identification and quantification of some potato blemishes in an acceptable time.

This system uses several image processing and machine learning techniques to automatically learn the appearance of different anomaly types.

It also incorporates an intuitive graphical user interface (GUI) handled with QT to enable easy set-up of the system by quality control (QC) staff working in the industry.

For further improvements of this system that includes adding 3D vision system, we need to have a basic version of TADD that works without dependencies on specific hardware setup, e.g. GPU. For this purpose, we have started modifications on two new machines:

- One PC has GPU and we used it to learn dependencies on other software packages (QT or OpenCV libraries): on this machine, we need to debug QT dependencies to get TADD working as most of the problems caused by QT functions which are: (1) QtCvImageWidget, (2) QtCvOverlayWidget, (3) QtTODD, (4) TODDengine and (5) CvCooccurrence as shown in the list of the functions in Figure 7.1.

- The other PC does not have GPU and we used it to cut dependencies on GPU processing mainly in glic superpixel algorithm Ren & Reid [2011]. For this purpose, we have another library for superpixel segmentation that works without any need of parallel processing within a reasonable time Achanta *et al.* [2012].

This whole process will end by end of the next quarter as we need to modify TADD software residing in our project partner Sutton Bridge Potato Council. Meanwhile, we have started modifying the current TADD to be fed with RGB-D (3D) data to compare 3D and 2D modes. This is the core topic of the next chapter.

7.3 3D TADD

We have investigated some 3D sensors (SICK and Cognex) to assess their capabilities for our project needs. However, none of them met our requirements mostly for user friendliness, calibration and sensitivity to noise. Until we find a suitable 3D vision sensor, we need to modify TADD software to work with RGB-D data. For this purpose, the work has started and one on hand solution to record 3D data is to use Microsoft Kinect with a zooming lenses to reduce minimum distance of the object to the camera. Kinect has an RGB camera and an infrared emitter and camera. They enable Kinect to capture colour image as well as depth of each pixel in the scene. However, the current Kinect suffers from not having a proper resolution that makes it difficult for precision tasks like blemish detection but it is suitable for volume analysis of food contents as shown in Figure 7.2. In this report, we used Microsoft Kinect SDK to record colour and depth images in Visual C++ environment. Due to low resolution of Kinect, we recored image data only for big vegtables like watermelon as shown in Figure 7.2[1] .

Current 2D TADD system incorporates some luminance, chrominance, edge statistics as described in Hutton [2012]. This list includes: 1. Mean Red 2. Standard dev. Red 3. Skewness Red 4. Mean Blue 5. Standard dev. Blue 6. Skewness Blue 7. Mean Green 8. Standard dev. Green 9. Skewness Green 10. Mean Normalised Red 11. Standard dev. Normalised Red 12. Skewness Normalised Red 13. Mean Normalised Blue 14. Standard dev. Normalised Blue 15. Skewness Normalised Blue 16. Mean Normalised Green 17. Standard dev. Normalised Green 18. Skewness Normalised Green 19. Mean Intensity 20. Standard dev. Intensity 21. Skewness Intensity 22. Mean Edge Intensity 23. Standard. Dev. Edge Intensity 24. Skewness Edge Intensity 25. Mean Range 26. Standard Dev. Range 27. Skewness Range 28. Sobel edge length 29. Sobel edge count 30. Range edge length 31. Range edge count

Further improvements can be achieved in 2D texture analysis part of the image by applying methods like adding grey-level co-occurrence matrices, Gabor filter banks or local auto-covariance. Moreover, by adding depth data to TADD we need to have a new functionality that is capable of 3D feature extraction. As an

[1]Dataset can be downloaded from: https://www.dropbox.com/sh/fmbjh95rxeiyc5h/6Ff9DQ1vae

example for potential 3D features, depth histograms are shown in Figure 7.3 where we can differentiate between multiple objects and discard background with simple depth thresholding. In the future months, we will be investigating and selecting 3D features that can add accuracy to TADD with a reasonable computational complexity. All the programming tasks will be performed in C++ using OpenCV libraries and QT designed interface. Furthermore, we need to use 3D information for empty food trays and then food trays with contents to see 3D profile variations. This is one of our objectvies during the next quarter.

7.4 Conclusions

In this January 2014 report, we tried to accommodate depth data to the current TADD software to create 3D-TADD as well as working on the old software to make it executable on any PC. For this purpose, the time schedule for January 2014 is as follows:

- 8-17 January: Debugging current TADD software in UoL on a fresh PC with GPU and one PC without GPU
- 18-23 January: Working with Microsoft Kinect SDK libraries to capture RGB-D data with Kinect in UoL
- 27-31 January: Due to multiple errors in both new machines for debugging TADD software, we have carried out coding on the working machine to accommodate depth data for classification
- Meetings in January are: 15 January at SICK UK, 24 January at Sutton Bridge Potato Council and 29 January at UoL with Ishida.

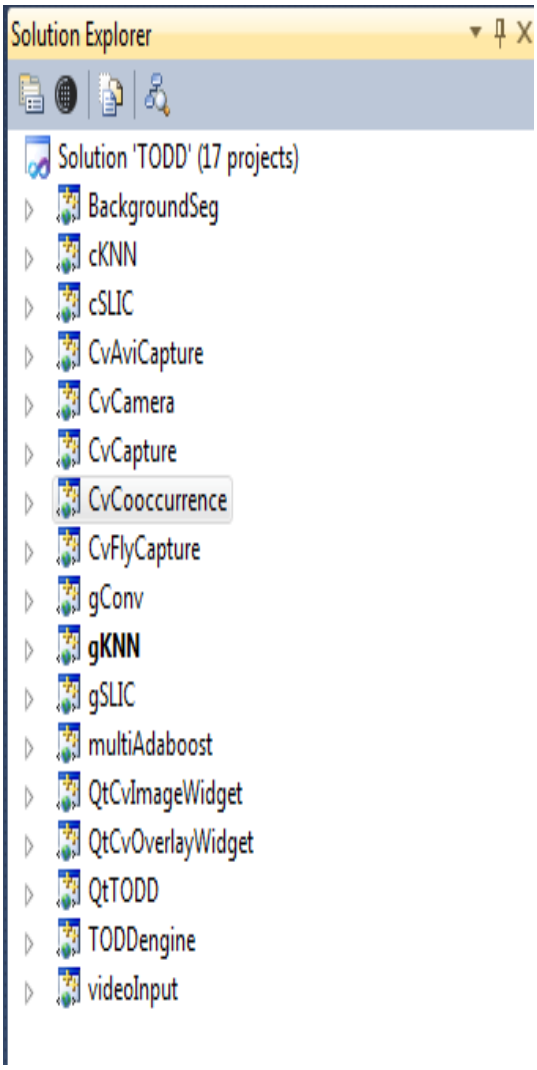


Figure 7.1: The list of the current TADD functions

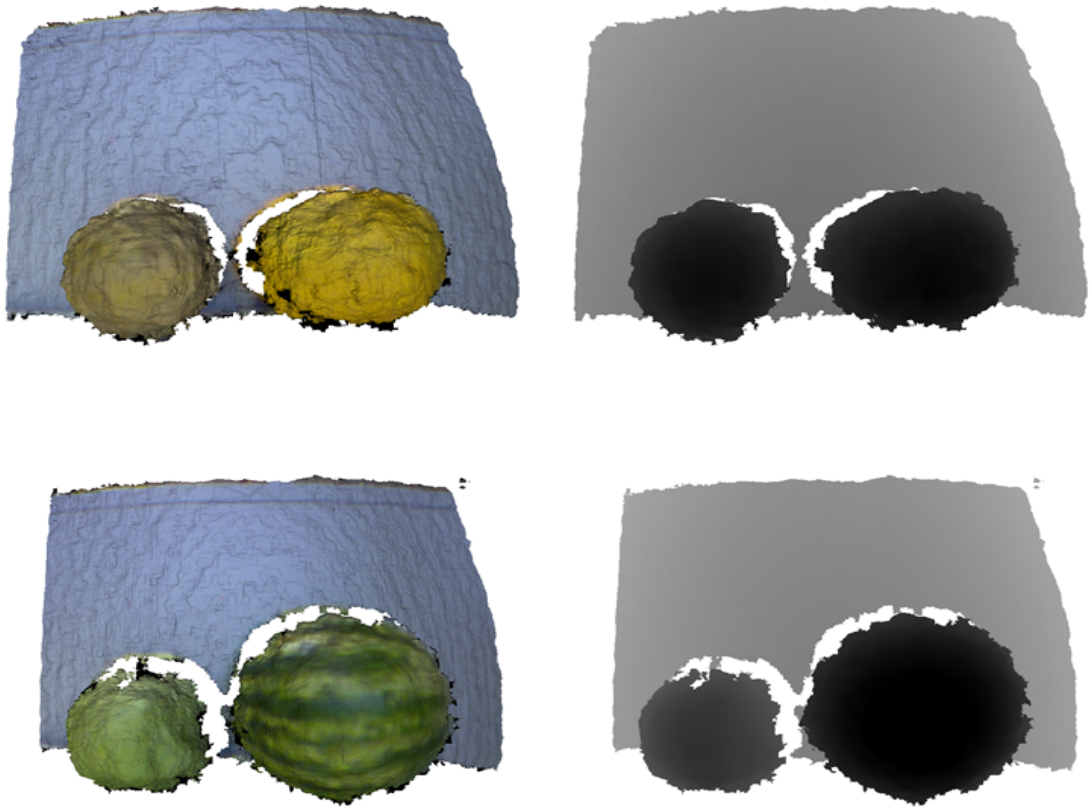


Figure 7.2: Side-by-side colour and inverse-depth images recorded with Kinect for big vegetables where closer objects are darker.

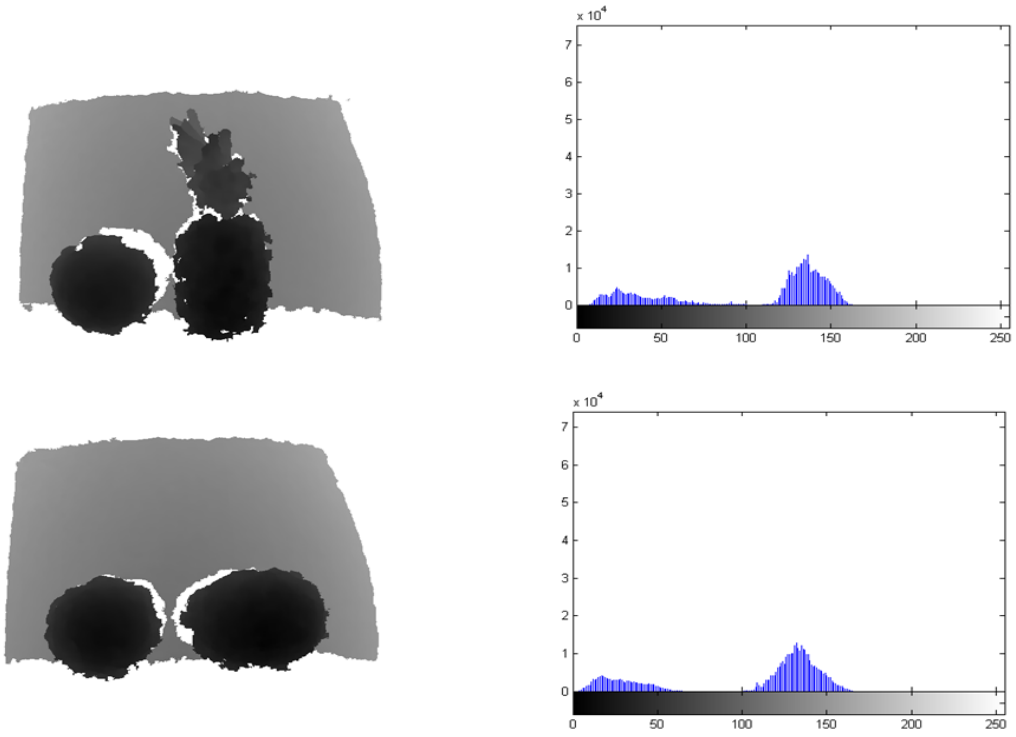


Figure 7.3: Inverse-depth maps and corresponding histograms recorded with Kinect for big vegetables. This data help us to discard background and discriminate between objects as they have different depth profiles.

References

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P. & Susstrunk, S. (2012). Slic superpixels compared to state-of-the-art superpixel methods. 41, 52

Bay, H., Tuytelaars, T. & Van Gool, L. (2006). Surf: Speeded up robust features. In *Proc. ECCV*, 404–417, Springer. 2, 11, 22

Breiman, L. (1993). *Classification and regression trees*. CRC press. 36

Breiman, L. (2001). Random forests. *Machine learning*, **45**, 5–32. 36, 37

Chen, M., Dhingra, K., Wu, W., Yang, L., Sukthankar, R. & Yang, J. (2009). Pfid: Pittsburgh fast-food image dataset. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, 289–292, IEEE. 41

Fang, L. & Xie, C. (2010). 1-d barcode localization in complex background. In *Computational Intelligence and Software Engineering (CiSE), 2010 International Conference on*, 1–3, IEEE. 11

Fischler, M.A. & Bolles, R.C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, **24**, 381–395. 2, 11, 22

Frankot, R.T. & Chellappa, R. (1988). A method for enforcing integrability in shape from shading algorithms. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **10**, 439–451. 32

Hutton, J. (2012). *A Prototype Low-Cost Machine Vision System For Automatic Identification and Quantification of Potato Anomalies*. Master's thesis, Lincoln School of Computer Sciense, University of Lincoln. 52, 53

Lowe, D. (2004). Distinctive image features from scale-invar-iant keypoints. *International Journal of Computer Vision*, **60**, 91–110. 2

Ng, A. (2013). Deep learning. v, 37

Ren, C.Y. & Reid, I. (2011). gslic: a real-time implementation of slic superpixel segmentation. *University of Oxford, Department of Engineering, Technical Report*. 52

Vedaldi, A. & Fulkerson, B. (2010). Vlfeat: An open and portable library of computer vision algorithms. In *Proceedings of the international conference on Multimedia*, 1469–1472, ACM. 42

Witten, I.H., Frank, E. & Hall, M.A. (2011). *Data Mining: Practical Machine Learning Tools and Techniques: Practical Machine Learning Tools and Techniques*. Elsevier. 35, 36

Woodham, R.J. (1980). Photometric method for determining surface orientation from multiple images. *Optical engineering*, **19**, 191139–191139. 31