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

May 2014

Contents

Contents i

List of Figures iii

1 Introduction 1

2 Ran Song TADD Technical Report for February 2014 2

2.1 Abstract 2
2.2 Introduction 2
2.3 An overview of the Tesseract OCR 3
2.4 Challenges 5
2.5 Our method 6
2.6 Conclusions 7

3 Ran Song TADD Technical Report for March 2014 12

3.1 Abstract 12
3.2 Introduction 13
3.3 Software architecture 14
3.4 Potential improvement and future work 15

4 Ran Song TADD Technical Report for April 2014 21

4.1 Abstract 21

5 Hossein Malekmohamadi TADD Technical Report for February 2014 22

5.1 Introduction 22

5.2 TADD Software Developments 23
5.3 Label-TADD: OCR integration 25
5.4 Conclusions 26

6 Hossein Malekmohamadi TADD Technical Report for March 2014 37
6.1 Introduction 37
6.2 QA-TADD Development 37
6.2.1 TADD Software Developments 37
6.2.2 Object Identification with HALCON 39
6.2.3 Conclusions 40

7 Hossein Malekmohamadi TADD Technical Report for April 2014 44
7.1 QA-TADD Development 44

List of Figures

2.1 Left: the ROI image output by our label detection algorithm; Right: the results of directly applying Tessearct OCR on it 4

2.2 Left: the synthetic ROI image; Right: the results of directly applying Tessearct OCR on it 5

2.3 The reference image that we used 6

2.4 Feature detection on the reference and the ROI images 8

2.5 A comparison of the feature matching with and without RANSAC where we shade the two input images in red and blue respectively for a better visualisation result. The yellow lines connect the pairs of matching points which correspond to the same position in the real world. Left: without RANSAC; Right: with RANSAC. 8

2.6 Left: the registered image; Right: the features points in the registered image are now fully overlapped with the ones extracted from the reference image, which means that the registration is very successful. 9

2.7 Left: the 'textness' map where warm colour denotes high textness; Right: the final output of our method is a correct detection of the one-line text that we are interested in. And we save the red rectangle as a digital image on the disk. 9

2.8 The one-line text is correctly recognised by Tessearct. 9

2.9 In this test, we used a reference image where the $4mm$ text line in the label was removed. 10

2.10 In this test, we used a reference image where the digits of 600 in central region of the label was removed. 11

3.1 The beginning piece of our C++ codes. 13
3.2 The workflows of the learning components of the software. Reference Image 1 is the image of a food tray without the label; Reference Image 2 is the image of a food tray where its label is at the correct position and with correct orientation; Reference Image 3 is the image of the label without the text line of interest. 16
3.3 Examples of the three reference images and the ROI image 17
3.4 The workflow of the first deployment component 18
3.5 The workflow of the second deployment component 19
3.6 Part of the codes designed to implement feature detection and extraction. 20
3.7 A piece of codes used to do thresholding for the ROI image 20
3.8 The current mechanism that we developed to input more than one test images 20

5.1 Quality report sample from Branston ltd 28
5.2 OCR results for the original image, filtered image and resized image for a part of a label (printed characters are all numbers) 29
5.3 OCR results for the original image, filtered image and resized image for a part of a label (printed characters are letters and numbers) . 29
5.4 OCR results for the original image, filtered image and resized image for a part of a label (printed characters are letters) 30
5.5 OCR results for the original image, filtered image and resized image for a part of a label (date is printed) 30
5.6 OCR results for the original image, filtered image and resized image for a part of a label (a brand name) 31
5.7 OCR results for the original image, filtered image and resized image for a part of a label (multiple lines of alphabets and numbers) . . 31
5.8 OCR demo of HALCON for a dot-printed date on a graphical background 32

5.9 Tesseract-OCR for a dot-printed date on a graphical background. It shows that for this kind of OCR application we need to put effort on image enhancement techniques, e.g. background segmentation or image morphology for the dot-printed area 32
5.10 Demo of photometric application in HALCON to detect surface defect in a shampoo container 33
5.11 An ideal text localisation where text regions are marked. We need an automated procedure to detect these regions and create sub-images from them before OCR. 33
5.12 Text regions are detected using Neumann & Matas [2012]. 34
5.13 Text regions are detected using Neumann & Matas [2012]. Comparing this example with Figure 5.12 shows that we need a robust text localisation algorithm that is not subject to small orientations. 34
5.14 Text regions are detected using Neumann & Matas [2012]. There are some false detections in the barcode region. 35
5.15 OCR results for the original image, filtered image and resized image for the output of Figure 5.14. 35
5.16 OCR results for a captured image applying web-based text localisation in http://www.textspotter.org/. 36
5.17 OCR results for a captured image applying web-based text localisation in http://www.textspotter.org/. The algorithm in http://www.textspotter.org/ shows robustness to orientation errors within a tolerance. 36

6.1 Pruning the project diagram 41
6.2 Example 1 of Halcon demo 42
6.3 Example 2 of Halcon demo 43

Chapter 1

Introduction

This technical quarterly report is a summary of the work we have done in February, March and April 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 February 2014

2.1 Abstract

In our last report, we introduced our method for label detection in a food tray image. The detected label region is essentially a region of interest in which the system will further implement the optical character recognition (OCR). This report will give you a brief background introduction to the typical procedures involved in the OCR process. Also, the challenges that we are facing at the current stage of the project are discussed, which are mainly caused by our specific needs in the integration of automatic label detection and reliable OCR from a complicated image background. Aiming at these challenges, we propose a teaching-by-showing method. It is inspired by the success of our label detection technique based on a similar teaching-by-showing process. Although the preliminary results are promising, we still need to make a great effort in the improvement the proposed method to make it more robust for the real industrial applications.

2.2 Introduction

We have already developed and demonstrated our label detection technique which can automatically and intelligently detect the label position and orientation. Nat-

urally, the next step is to apply some OCR to read the texts within it. Note that the word 'read' here means the transfer from the appearance and the shapes of the characters to the ASCII codes corresponding to them. The ASCII codes actually form a string in a computing language (e.g., MATLAB or C++) which can be easily used in the direct comparison with the correct information (such as date, weight, price, size, etc.). In this way, the system can recognise the tray with a label containing incorrect information.

The rest of the report is organised as follows. In Section 2, we briefly introduce the OCR technique employed in our method. In Section 3, we list the difficulties that need to be conquered to incorporate the OCR into our label-TADD system. In Section 4, we propose our own method to solve the problem and show some preliminary results. Finally, we conclude in Section 5.

2.3 An overview of the Tesseract OCR

Tesseract is an open-source OCR engine developed by HP. It was released in late 2005 and is now available at http://code.google.com/p/tesseract-ocr. In Tesseract, the first step is a connected component analysis in which outlines of the components are stored. At this stage, outlines are gathered together, purely by nesting, into *Blobs*.

Blobs are then organised into text lines, and the lines and regions are analysed for fixed pitch or proportional text. Text lines are broken into words differently according to the kind of character spacing. Fixed pitch text is chopped immediately by character cells. Proportional text is broken into words using definite spaces and fuzzy spaces.

Next, recognition proceeds as a two-pass process. In the first pass, an attempt is made to recognise each word in turn. Each word that is satisfactory is passed to an adaptive classifier as training data. The adaptive classifier then gets a chance to more accurately recognise text lower down the page. Since the adaptive classifier may have learned something useful too late to make a contribution near the top of the page, a second pass is run over the page, in which words that were not recognised well enough are recognised again.

A final phase resolves fuzzy spaces, and checks alternative hypotheses for the

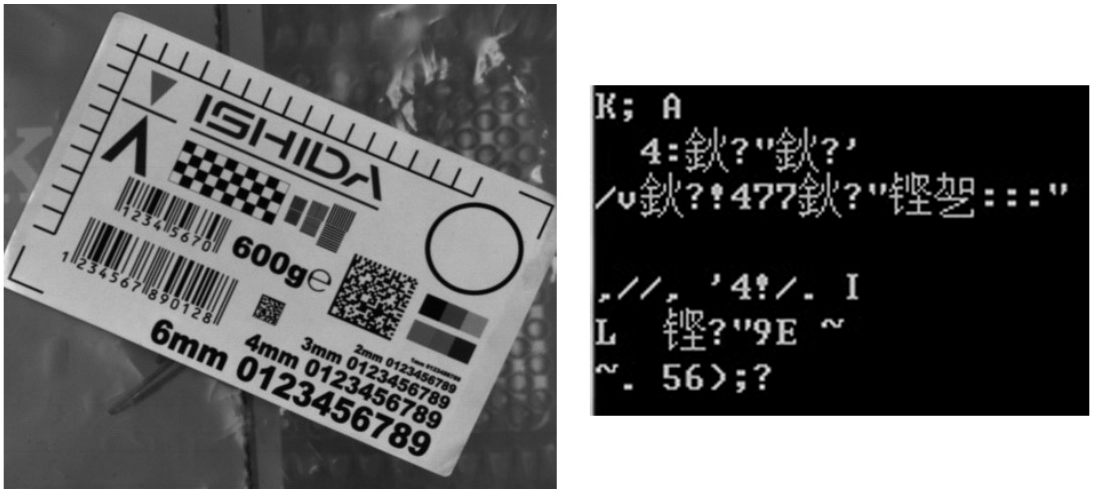


Figure 2.1: Left: the ROI image output by our label detection algorithm; Right: the results of directly applying Tessearct OCR on it

x-height to locate small-cap text.

In Tesseract, since the classifier is able to recognise damaged characters easily, the classifier was not trained on damaged characters. In fact, its classifier was trained on a mere 20 samples of 94 characters from 8 fonts in a single size, but with 4 attributes (normal, bold, italic, bold italic), making a total of 60160 training samples. However, Tesseract contains relatively little linguistic analysis. This could cause potential problems in our applications since the texts on a food tray could be in French, German, Dutch, Welsh, etc. In this project, we do not intend to make any effort on that since it is out of the scope. In the demonstration, we shall only use English texts. The fact is that there is no generic OCR solution due to the variety of languages. For instance, the OCR algorithms for some oriental languages such as the Chinese are quite different from those for Latin languages because the graphical structures of the Chinese characters are significantly different. In practice, we can always use different OCRs to handle text lines with different languages.

Figure 2.2: Left: the synthetic ROI image; Right: the results of directly applying Tessearct OCR on it

2.4 Challenges

We show Fig. 2.1 to illustrate the challenges involved in our work. The left image shown in Fig. 2.1 is the output region-of-interest (ROI) image delivered by our label detection method. Instead of a rotated bounding box, we save the horizontal bounding box of the label region as an image on the disk since an image must be a horizontal rectangle. In this way, we can use the ROI image as the input for the Tesseract OCR. Unfortunately, as shown in the right image, it failed to read the text lines in the label. It is not hard to realise that the failure could be caused by the non-text patterns within the label since they can make the OCR 'confused'. Thus we made a synthetic image as shown in Fig. 2.2 where we removed the non-text part of the original ROI image. And we ended up with the right image of Fig. 2.2. Certainly, we can observe some improvements but clearly we cannot claim that the OCR is successful here.

The reason behind these failures is that the Tesseract OCR does not have its own page layout analysis, also known as text localisation techniques. Tesseract assumes that its input is a binary image with optional polygonal text regions defined because HP had independently-developed page layout analysis/text localisation technology that was used in products (and therefore not released for open-source). Also, to the best of our knowledge, like most OCR techniques, Tesseract is not very robust to rotation.

Text localisation itself can be a very complicated problem if in particular, there is no learning information. Because of its wide range of applicability, some IT giants such as Google is funding some people to try to solve it in a generic manner

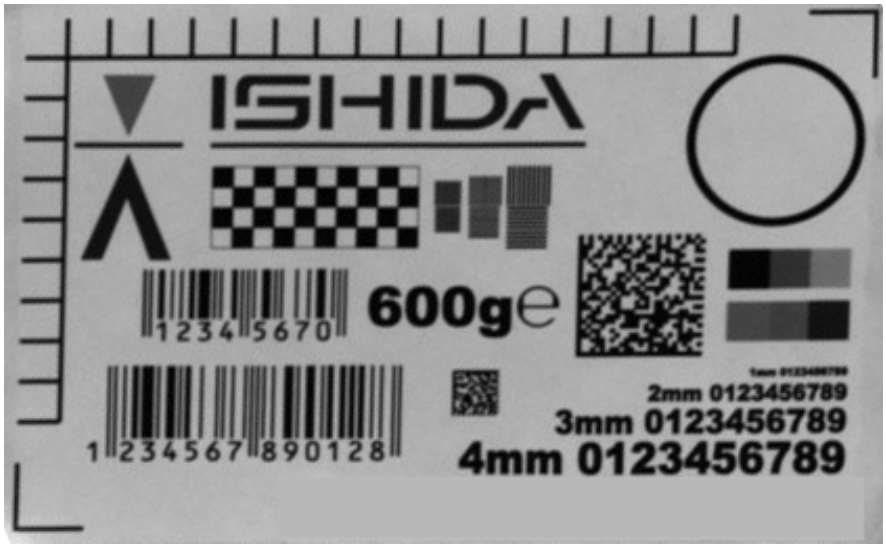


Figure 2.3: The reference image that we used

(http://cmp.felk.cvut.cz/ neumalu1/). There is no point for us to compete with Google on this trend. Thus, we propose a teaching-by-showing method which focuses specifically on our food tray application.

2.5 Our method

The teaching-by-showing strategy has been successfully applied for label region detection. Please refer to our previous reports for details.

In the proposed method, first, we need a reference image where the targeted text line is removed and the label in it is roughly horizontal. Fig. 2.3 shows an example. In practice, such a reference image can be a standalone one (e.g., a printing template) loaded from the disk or detected via our label detection method within a food tray image. Note that in this synthetic image, we deliberately remove the text line at the bottom and use it as the reference image.

We first input this reference image and the real ROI image shown in Fig. 2.1 and implement the SURF feature detection Bay *et al.* [2006] on both of them. The results are shown in Fig. 2.4. The following algorithms are similar to our label detection method, including feature matching, RANSAC-based refinement, transform estimation, image warping, textness map generation and text localisation.

Therefore, in this report, we do not describe them in detail again. Figs. 2.4–2.7 visualise the results of the aforementioned algorithms step by step. Fig. 2.8 shows that the output of our method—an image saved on the disk which merely contains the one-line text can be easily and correctly recognised by the Tessearct. It is worth pointing out that compared with the previous labet detection method, we make some changes in the textness map generation step. For instance, the erosion and dilation of the difference image is now implemented on a smaller scale.

Figs. 2.9 and 2.10 show more results using different test images and reference images.

2.6 Conclusions

Note that the current version of our text localisation method is based on MATLAB since we just want to check whether our new method can work and work reliably. If it performs really well over a variety of test data, we shall consider further developing it to a full C++ implementation in the next stage. Although the results shown in this report is very promising, it is not that difficult to see that it could be not that robust for some labels without enough patterns. Since the proposed method is feature-dependent, its reliability is highly related to the number of features extracted in the region of interest. However, a label on a food tray could be very simple and not textured enough. Hence we still need to further test the current method and see whether we can make some improvement in the near future. In short, efficiency seems not a big problem since the ROI image of a label is typically small; meanwhile, we will concentrate more on the reliability of the method.

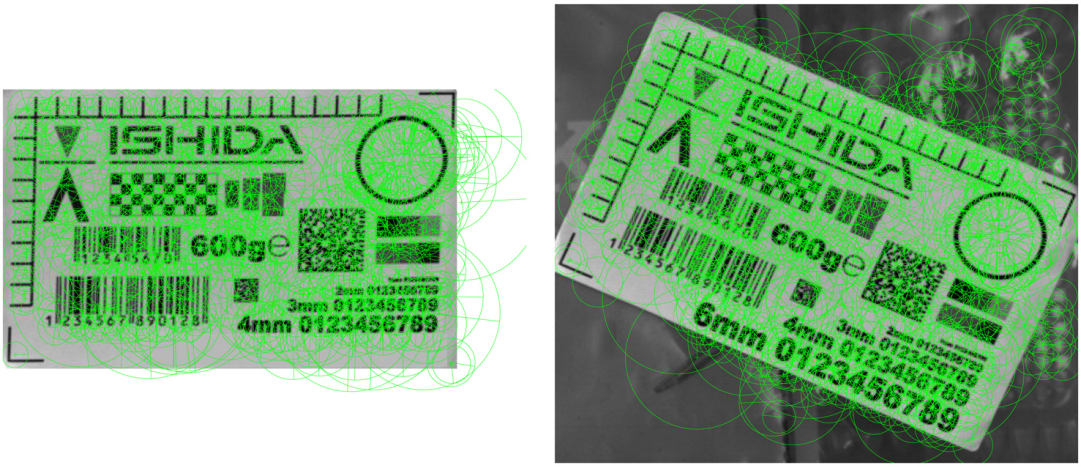


Figure 2.4: Feature detection on the reference and the ROI images

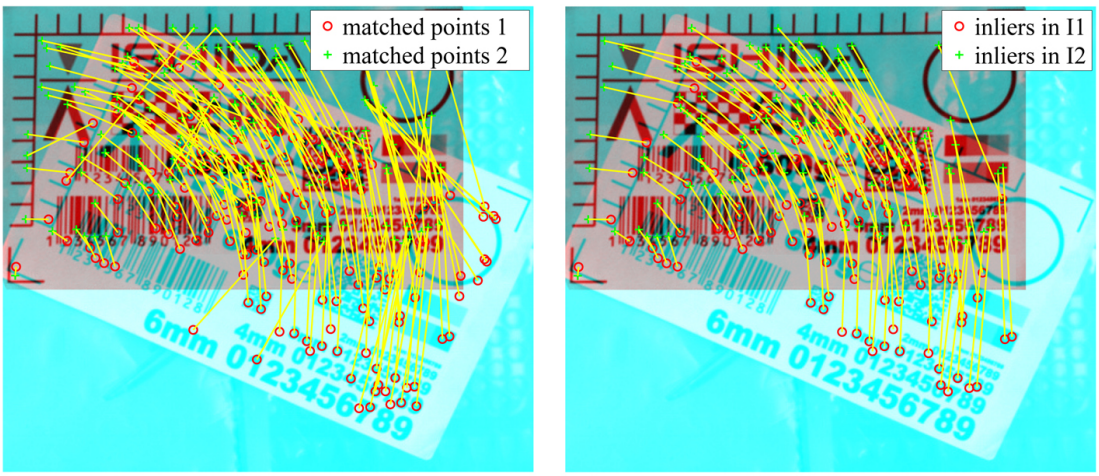


Figure 2.5: A comparison of the feature matching with and without RANSAC where we shade the two input images in red and blue respectively for a better visualisation result. The yellow lines connect the pairs of matching points which correspond to the same position in the real world. Left: without RANSAC; Right: with RANSAC.

Figure 2.6: Left: the registered image; Right: the features points in the registered image are now fully overlapped with the ones extracted from the reference image, which means that the registration is very successful.

Figure 2.7: Left: the 'textness' map where warm colour denotes high textness; Right: the final output of our method is a correct detection of the one-line text that we are interested in. And we save the red rectangle as a digital image on the disk.

Figure 2.8: The one-line text is correctly recognised by Tessearct.

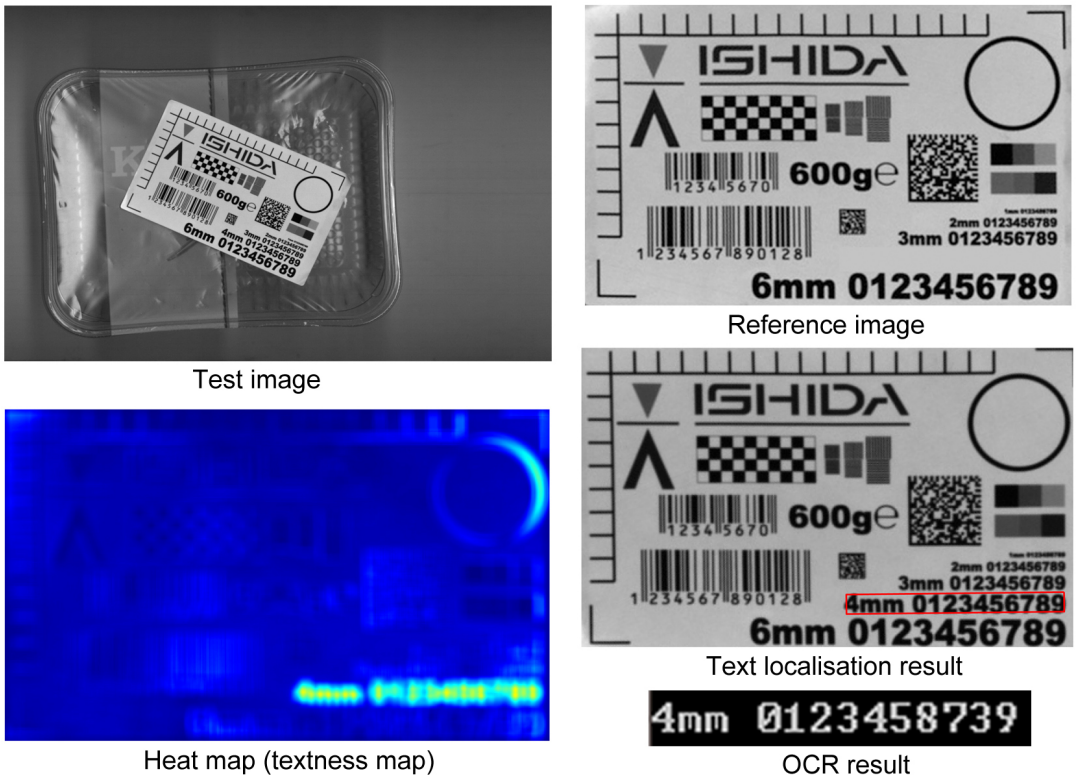


Figure 2.9: In this test, we used a reference image where the $4mm$ text line in the label was removed.

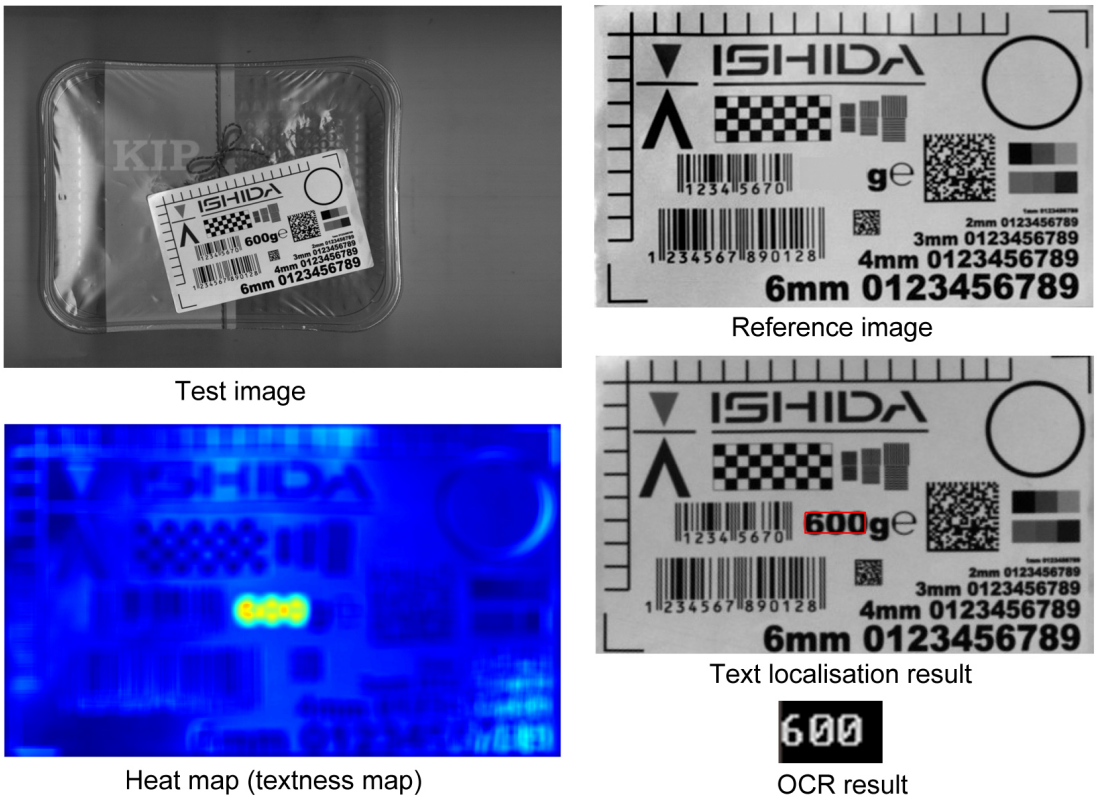


Figure 2.10: In this test, we used a reference image where the digits of 600 in central region of the label was removed.

Chapter 3

Ran Song TADD Technical Report for March 2014

3.1 Abstract

In our last report, we extended previous work on label detection by introducing new techniques for text localisation and optical character recognition (OCR). The whole algorithm was implemented in MATLAB. The major advantage of MATLAB is that it is relatively simple since most of its functions are of high level and sometimes a complicated image processing algorithm can be implemented in merely several lines of MATLAB codes. Therefore, it is good at fast development of new algorithms, testing the correctness or feasibility of a new algorithm and observing the preliminary results, etc. However, it is inefficient and inconsistent with the industrial standard for software development. More importantly, it is not standalone, requiring the installation of MATLAB components in the computing devices of end users. These components, such as MATLAB Compiler Runtime (MCR), are typically not free. In March, we further redevelop the existing algorithms for automatic label detection, intelligent text localisation and OCR in the standard industrial computing language for software development — C++. It is more efficient, and fully standalone. More importantly, the entire development is based on an open source computer vision library, OpenCV and thus it is completely free.

```
#include <stdio.h>
#include <iostream>
#include "opencv2/core/core.hpp"
#include "opencv2/features2d/features2d.hpp"
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/calib3d/calib3d.hpp"
#include "opencv2/nonfree/nonfree.hpp"
#include "opencv2/video/tracking.hpp"
//#include <time.h>

using namespace std;
using namespace cv;

int main( int argc, char** argv )
{
     if( argc != 4 )
  { return -1; }

  Mat img_object= imread(argv[1], CV_LOAD_IMAGE_GRAYSCALE );
  Mat img_ref= imread(argv[2], CV_LOAD_IMAGE_GRAYSCALE);
  Mat img_scene= imread(argv[3], CV_LOAD_IMAGE_GRAYSCALE);

  if( !img_object.data || !img_ref.data||!img_scene.data )
  { std::cout<< " --(!) Error reading images " << std::endl; return -1; }
```

Figure 3.1: The beginning piece of our C++ codes.

3.2 Introduction

In the TADD project, the paradigm of our software development process is to first implement a novel idea using MATLAB to see whether it can work and how well it works due to MATLAB's advantage of rapid development as is mentioned in the abstract. Then, if the result is good (guaranteeing the algorithmic/technical correctness), we shall consider using C++ to reimplement the algorithm for efficiency reasons and making sure that it is standalone. This is just what we are doing in March. Different from the existing software installed in the Ishida machine where Sherlock is the dominant third-party machine vision library, the redevelopment using C++ in our TADD project is based on OpenCV.

OpenCV is a library of programming functions mainly aimed at computer vision, developed by INTEL, and now supported by Willow Garage and Itseez. It is released under a BSD license and hence it is free for both academic and

commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. The reason that we choose OpenCV for our software development in TADD is that it was designed for computational efficiency and with a strong focus on real-time applications. Written in optimised C/C++, the library can take advantage of multi-core processing. It is a well-documented and widely-supported library with more than 47 thousand people of user community and estimated number of downloads exceeding 7 million. In the TADD project, we use its 2.4.8 version to facilitate our software development.

3.3 Software architecture

The codes are based on a couple of OpenCV built-in functions. Figure 3.1 shows the beginning piece of the codes. It can be seen that we utilise the OpenCV functions lying in 6 categories, namely core, features2d, highgui, calib3d, nonfree, and tracking. In our software, these functions provide us the optimised implementation of feature extraction, feature matching, refinement of matches, feature-based motion estimation, image erosion and dilation, image thresholding and some other trivial computer vision algorithms. In this section, we summarise and document the main architecture of the codes, which does not only make a clear view for the better understanding of our software, but will also benefit the potential improvement and updating in the future work.

The software can be separated into two types of components, learning and deployment. The two learning components are shown in Figure 3.2. They are designed for teaching the system 'what is correct' and thus only need to be done once. Figure 3.3 (a)-(c) show three images as examples of the reference images required in the learning components. In a real application, if these reference images are updated, we need to redo the learning. In the first learning component, the output is the correct label position and orientation. Such information will be used in the comparison with the information describing the position and orientation of the label detected in a test image. The label detection in a test image is implemented in the first deployment component. Its outcome includes a judgement on the acceptance of tray for the following OCR and a region of interest (ROI) image saved on the disk (if the tray is rejected, there is no need to output

the ROI image). Figure 3.3 (d) gives an example of an ROI image. The second deployment component is designed for text localisation. Its output is an image with the text line of interest only. Then this text line image will be processed and input into the Tessearct OCR to output an ASCII String. Typically, the string denotes the information of the food product inside the tray, such as date, price and weight, etc.

To have a better understanding of the software development based on C++ using OpenCV, we show several lines of the codes in Figure. 3.6. These lines of codes are used to detect and extract the SURF features Bay *et al.* [2006].

3.4 Potential improvement and future work

It can be seen that the current system is an integration of several components. Therefore, reliability is crucial since every mistake made by a certain component will lead to a failure of the entire system. In particular, the image thresholding techniques required for the binarisation of the difference images are relatively sensitive to parameter settings. On the one hand, the variation of the contents of test images means that ideally, we need to tune the parameters every time in order to find the best threshold. On the other hand, our end users are not experts on image processing and it could be time consuming to fine tune the parameters in practice. We currently employ a direct thresholding. As shown in Figure. 3.7, the thresholding is dependent on the mean value of the image. Hence, the future work will focus on the employment and/or the development of some adaptive image thresholding techniques.

In addition, as shown in Figure. 3.8, in the current software, we rely on the command (1 or Ctrl+C) input by the user to decide whether the system should go on running. Thus another future work will be to make our software communicate with the machine vision camera mounted on the Ishida machine in an efficient and automated way. So the system can automatically decide by itself whether and when it should go on handling images.

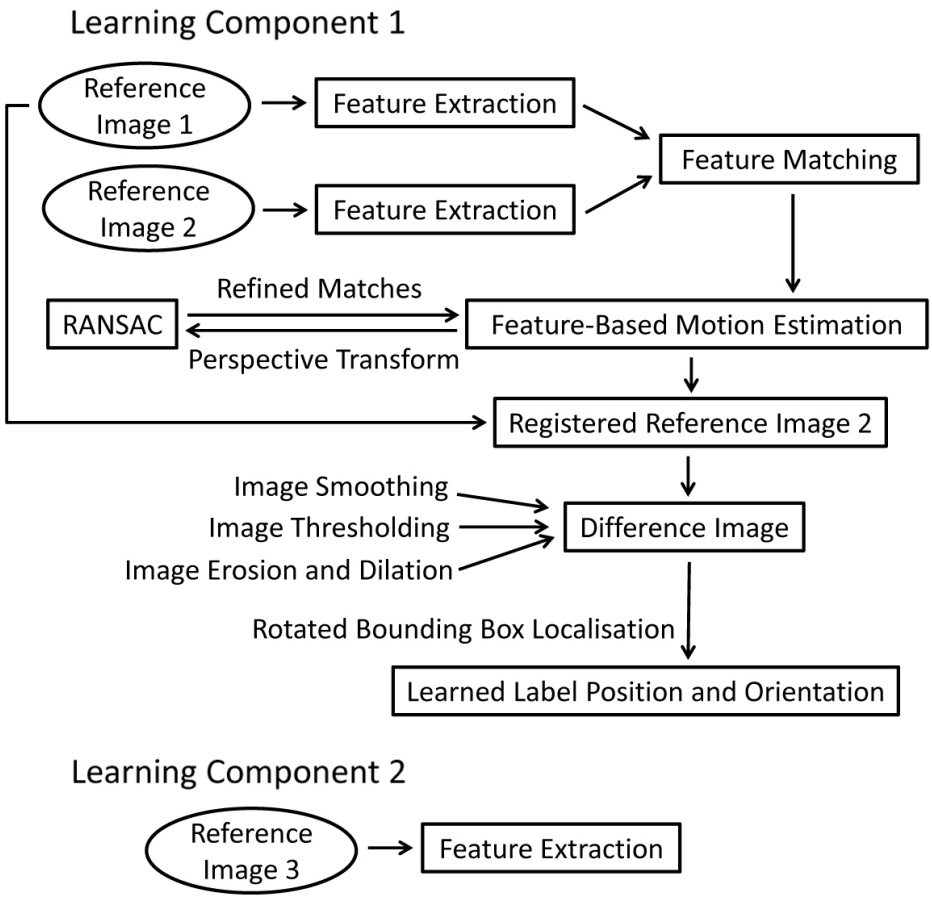
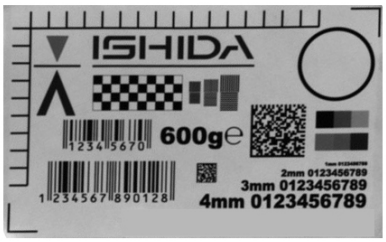


Figure 3.2: The workflows of the learning components of the software. Reference Image 1 is the image of a food tray without the label; Reference Image 2 is the image of a food tray where its label is at the correct position and with correct orientation; Reference Image 3 is the image of the label without the text line of interest.

(a) Reference Image 1



(c) Reference Image 3

(b) Reference Image 2

(d) ROI Image

Figure 3.3: Examples of the three reference images and the ROI image

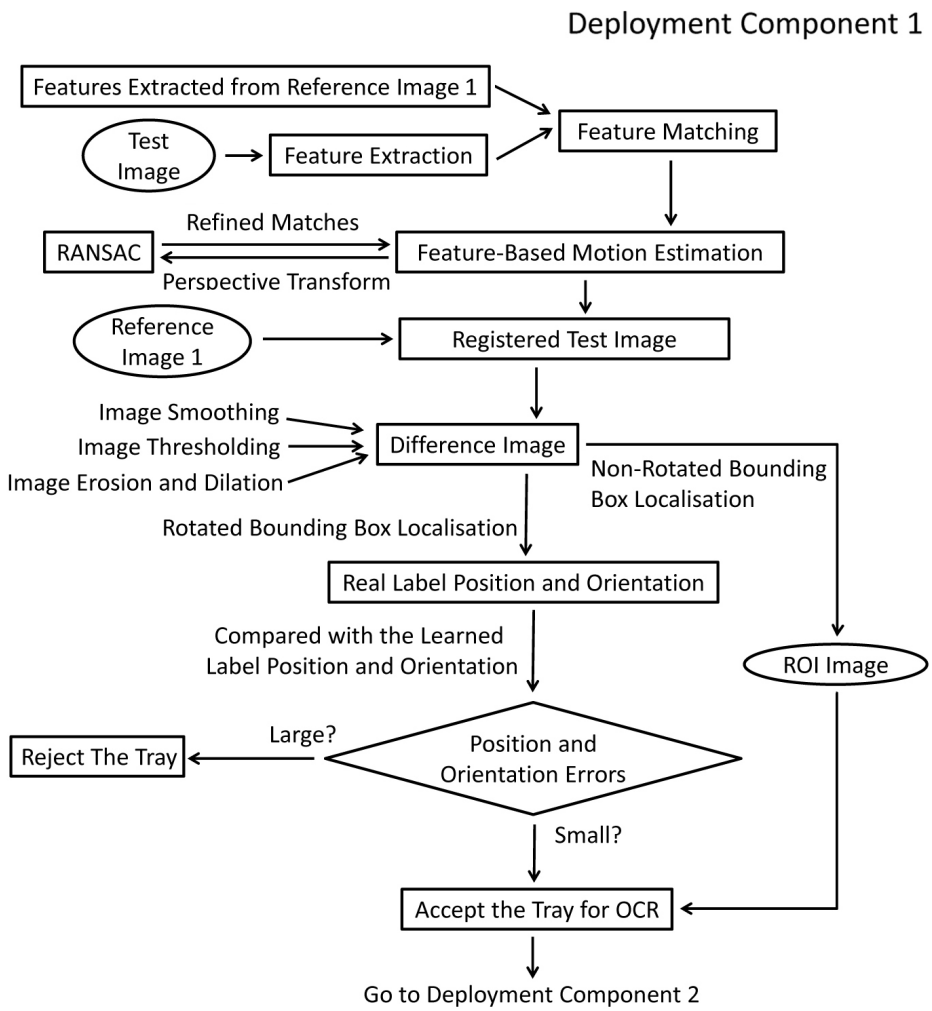


Figure 3.4: The workflow of the first deployment component

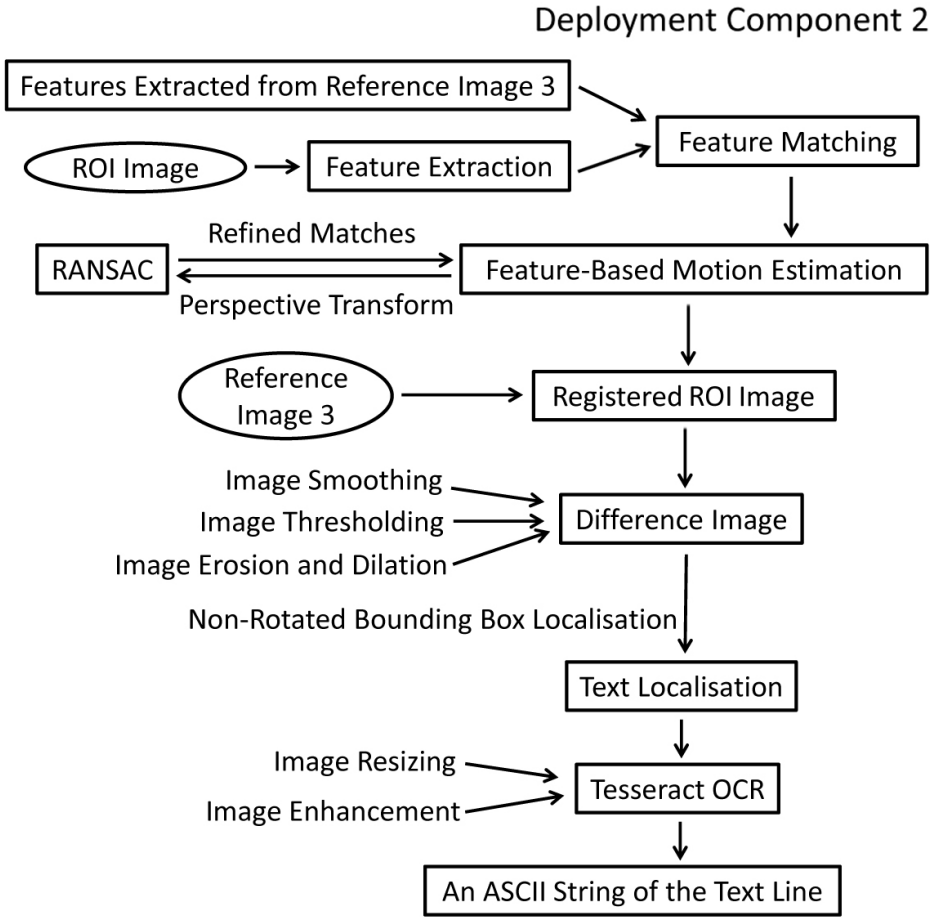


Figure 3.5: The workflow of the second deployment component

```
//-- Step 1: Detect the keypoints using SURF Detector
int minHessian = 400;
SurfFeatureDetector detector(minHessian);

vector<KeyPoint> keypoints_object, keypoints_ref, keypoints_scene, keypoints_roi, keypoints_extroi;
detector.detect(img_object, keypoints_object );
detector.detect(img_ref, keypoints_ref );

//-- Step 2: Calculate descriptors (feature vectors)

SurfDescriptorExtractor extractor;

Mat descriptors_object, descriptors_ref, descriptors_scene,descriptors_roi,descriptors_extroi;

extractor.compute( img_object, keypoints_object, descriptors_object );
extractor.compute( img_ref, keypoints_ref, descriptors_ref );
```

Figure 3.6: Part of the codes designed to implement feature detection and extraction.

```
Scalar meanthreshroi=mean(blur_roi);
float roithresh=2.5*meanthreshroi.val[0];

threshold(blur_roi,bw_roi,roithresh,255,THRESH_BINARY);
```

Figure 3.7: A piece of codes used to do thresholding for the ROI image

```
cout<<"Continue?"<<endl;
cout<<"1: Yes; Ctrl+C: Quit"<<endl;
cin>>isContinue;
if (isContinue==1)
    cout<<"Input a new test image"<<endl;
char Filename[20];
cin>>Filename;
img_scene= imread(Filename, CV_LOAD_IMAGE_GRAYSCALE);
```

Figure 3.8: The current mechanism that we developed to input more than one test images

Chapter 4

Ran Song TADD Technical Report for April 2014

4.1 Abstract

In April 2014, I worked with other members of the TADD team at Lincoln to write the paper titled 'Semantic Analysis of Food Images via Interactive Training and Classification'. This joint work has been submitted to BMVC 2014. We also made a video as the supplemental material attached to the submitted paper, which can now be watched at YouTube:

http://www.youtube.com/watch?v=fya4KQVUMQA (If 1080p high resolution can not be accessed in IE, please use Chrome browser).

The full paper is attached to this report.

Chapter 5

Hossein Malekmohamadi TADD Technical Report for February 2014

5.1 Introduction

This is the February 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 2013 report, we gave a brief introduction to a possible 3D vision system and decision forests. In the December 2013 report, we created training and test datasets for semantic segmentation for further research based on multiple datasets. In the January 2014 report, the aims were investigating 3D recording devices and extracting 3D features to analyse food contents. However, with the advent of the updated requirements

after the quarterly meeting in the early February 2014, the new priorities for this quarter are as follow:

- Creating the Github repositories for both the current QA-TADD in UoL and SB-TADD.
- Investigating HALCON capabilities for this project
- Investigating open-source OCR libraries for this project and integrate it to our current Label-TADD.
- Making a universal QA-TADD w.r.t to current QA-TADD in UoL and SB.

In this report, we cover all topics above.

5.2 TADD Software Developments

For a universal approach in software development of TADD, we needed to create repositories for previous software versions. Consequently, we came to this point to upload an document UoL-TADD and SB-TADD in Github and keep them as private repositories in LCAS-UoL members. The url for UoL-TADD (TADD-V1) and SB-TADD (TADD-V2) is: https://github.com/LCAS/TADD-project. The readme file for this project states that, the required Libraries are:

1. OpenCV 2.3.1
2. Qt 4.7.3
3. Boost 1.47
4. TBB (Intel Threading Building Blocks) $tbb40_20120408oss$
5. CUDA 4.0 (will build with 4.1)

TADD modules are:

1. QtTODD (main, startup project)

(a) QtCvImageWidget (paints capture frame to widget)

(b) QtCvOverlageWidget (paints overlay and accepts input)

(c) TODDengine (90% of processes are done here)

  i. CvCamera (wrapper for different capture methods)

    A. CvCapture (capture from standard webcam)

    B. videoInput (library used to try control exposure and gain)

    C. CvAviCapture (using pre-recorded as input)

    D. CvFlyCapture (point grey camera 640x 480)

  ii. cSLIC (CPU parts of SLIC algorithm)

    A. gSLIC(GPU parts of SLIC algorithm)

  iii. MultiAdaboost (Adaboost algorithm, training and learning)

And finally new items for TADD-V2 are:

1. User can assign class labels
2. New load and save features for classifier
3. Zoom to select feature in UI
4. Sliding window in UI
5. HD-Camera API integration in EDSDK library (http://www.usa.canon.com/cusa/consumer/standard/display/sdk/homepage)

As stated earlier, one of the main objectives for this current quarter is to unify software versions and have a QA-TADD with proper documentations and recommendations to be installed in any PC with a GPU. Moreover, we have started hardware setup with the help of a technician to mount a high resolution camera to SB-TADD rather than an off the shelf webcam for quality assessment of potatoes. We have started to replace OpenCV 2.3.1 with the latest version of OpenCV (2.4.8) because there are some feature extraction techniques which can be found in them, e.g. Local binary patterns (LBP). Other superpixel features that we can include are: 1) colour histogram, 2) dilated colour histogram, 3)

colour thumbnails or 4) sift clusters Tighe & Lazebnik [2010]. Finally, our aim is to cover QA needs of our project partners as much as possible. One example of the QA report is shown in Figure 5.1. A generic specification for QA can include:

1. Reject at 5% critical defects (pest damage, bruising, rots, internal defects e.g. Hollow Heart)
2. Reject at 10% mechanical damage
3. Reject at 30% skin finish, greens, (visual defects)

The above percentages are a guide for the Intake QA team in Branston ltd and can be challenged and if needed. The dimensions of the tray used in QA are:

- Height: 70mm
- Length: 600mm
- Width: 400mm

5.3 Label-TADD: OCR integration

In this chapter we have used Tesseract-OCR (https://code.google.com/p/tesseract-ocr) to read characters in digital images using C++ programming. This library has Apache License v2.0 (http://www.apache.org/licenses/LICENSE-2.0). For this purpose, we gathered data from multiple situations, e.g. date, label,... Figures 5.2 to 5.7 show the results for the original captured image, filtered image (denoised and threshold truncated image) and resized image (double size). For filtering, we have used OpenCV functions in C++ environment.

Meanwhile, we have investigate MVTec HALCON (http://www.halcon.com/). This software is not open source but it outperforms in cases like Figure 5.8 when we have a dot-printed character on a graphical label. This example is taken from HALCON demo software. we have used this digital image to test the open source OCR as shown in Figure 5.9 where it shows that we need to put effort on image enhancement techniques, e.g. background separation. In the demo version, there are few more interesting use cases like: photometric stereo in shampoo container

defect detection as shown in Figure 5.10 or leather surface investigation which can be converted and used in TADD applications for quality inspections.

Finally, another problem in using OCR is how to detect text regions. We need to develop a package in C++ for this purpose and divide the captured image to multiple sub-images containing text. Then we feed each sub-image to OCR. An example of ideal text localisation is shown in Figure 5.11. We show other examples of text localisation based on Neumann & Matas [2012] in Figures 5.12 to 5.15. A very good resource for this challenge can be found in http://cmp.felk.cvut.cz/~neumalu1/ and http://www.textspotter.org/. Another example of web-based text localisation based on http://www.textspotter.org/ is shown in Figures 5.16 and 5.17.

Having said these, the final suggestion for Label-TADD is:

1. Detect label region an compare it with reference image
2. Reject packs with position/orientation errors more than a threshold
3. For passed data, perform text localisation and output sub-images containing text
4. Perform image enhancement on each sub-image
5. OCR on each filtered sub-image

5.4 Conclusions

In this February 2014 report, the covered objectives in the TADD projects are:

- Attending quarterly review meeting in UoL and visiting Branston ltd in Lincoln.
- Software development stage for both current TADD and SB-TADD including setting up repository, installing new high resolution camera and documenting both projects.

- Testing open source OCR on several captured images using the Ishida machine in UoL, this part of the project is ready for integration to the Label-TADD.

- Investigating HALCON capabilities for the future plan of the project. Where we came across to the point that HALCON outperform open source softwares in some instances like dot-printed OCR on graphical background, e.g. printed date on yoghurt lid.

- Investigating some pre-existing techniques for text localisation to be implemented in the Label-TADD.

Branston Lincoln **Quality Report**

Sample Type	Intake	Forward	Routine Cold Store	Hot Box	Store Loading

Stock assessed as suitable for:

Tesco	Waitrose	Booker	Value	Reject

Grower	R Lindley Ltd	**Date Assessed**	24/02/2014
Variety	Melody	**QA**	J.Venkova
Field Ref	Alan Wilson	**Consignment no.**	147225
Store Ref		**Sample Temp °C**	10
Box ID		**Dry Matter %**	22.2

Recommend packing into:

Tesco	Waitrose	Booker	Value	Reject

Total pick-off 25 %

Size Analysis		
Whites	Weight	%
< 45mm		0
45-65	11.755	65
65-85	6.32	35
		0
85+		0
Total Wt	18.08	kg
Tesco Wt	18.08	kg

Defects	Weight	Critical Defect Wt	
Bruising		0.5	Total wt
Rots			0.5
Pest Damage			3%
Mech. Damage		0.645	
Greens		0.86	
Sprouting			
Skin Finish		2.435	
Growth Cracks			
Total weight of defects		4.44	kg

Comments

Internal Defects (10 tubers)

None	Minor	Severe
10	0	0

Skin Finish Key

1 = Level 1	2 = Level 2
3 = Level 3	4 = Level 4
Scab	1 to 2
Silver Scurf	3 to 4
Black Scurf	
Skin Spot	
Netting	3 to 4

Figure 5.1: Quality report sample from Branston ltd

Figure 5.2: OCR results for the original image, filtered image and resized image for a part of a label (printed characters are all numbers)

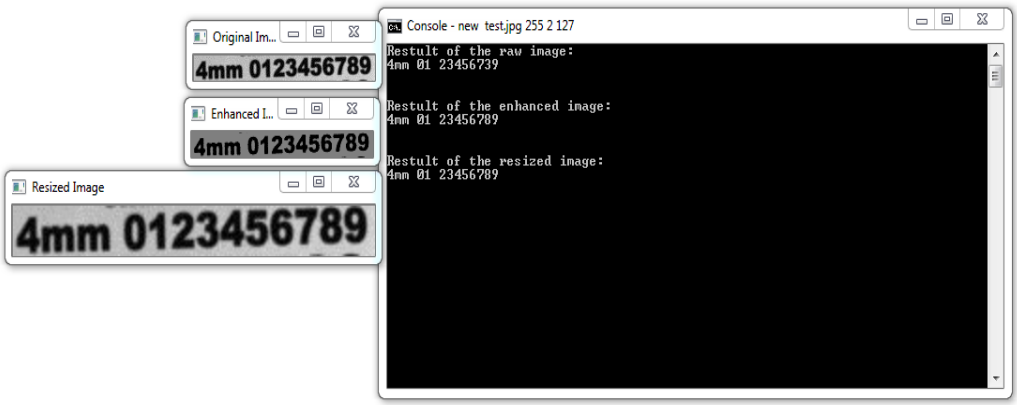


Figure 5.3: OCR results for the original image, filtered image and resized image for a part of a label (printed characters are letters and numbers)

Figure 5.4: OCR results for the original image, filtered image and resized image for a part of a label (printed characters are letters)

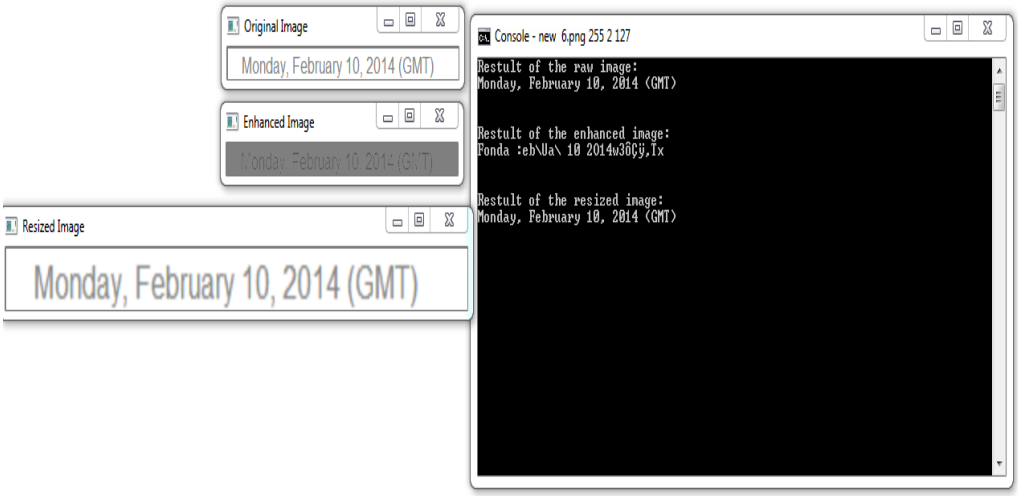


Figure 5.5: OCR results for the original image, filtered image and resized image for a part of a label (date is printed)

Figure 5.6: OCR results for the original image, filtered image and resized image for a part of a label (a brand name)

Figure 5.7: OCR results for the original image, filtered image and resized image for a part of a label (multiple lines of alphabets and numbers)

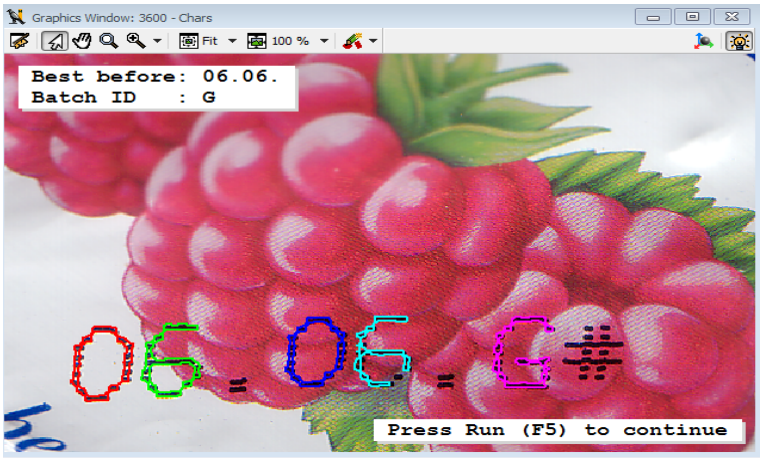


Figure 5.8: OCR demo of HALCON for a dot-printed date on a graphical background

Figure 5.9: Tesseract-OCR for a dot-printed date on a graphical background. It shows that for this kind of OCR application we need to put effort on image enhancement techniques, e.g. background segmentation or image morphology for the dot-printed area

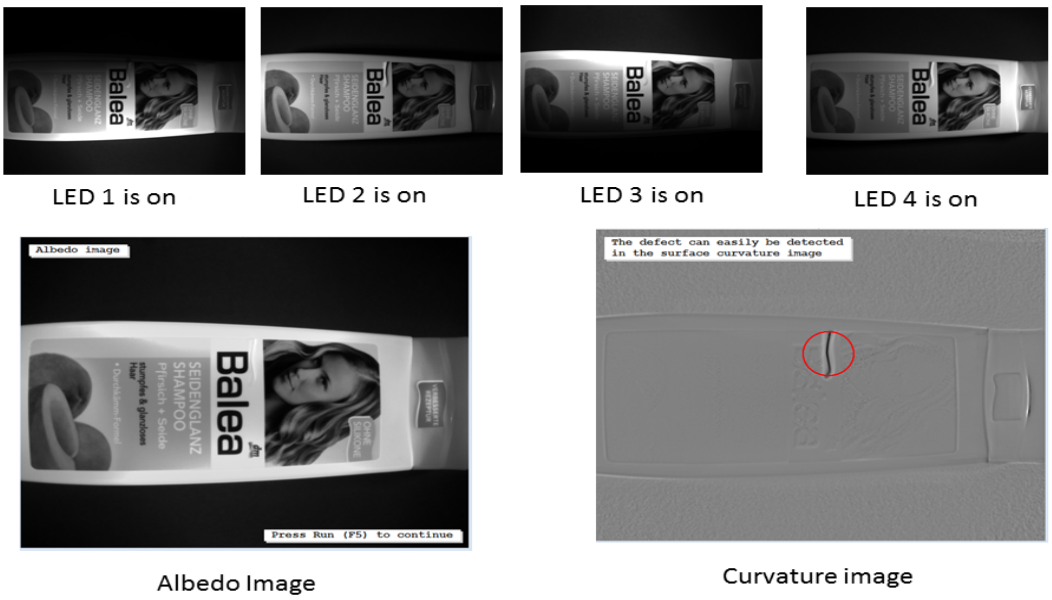


Figure 5.10: Demo of photometric application in HALCON to detect surface defect in a shampoo container

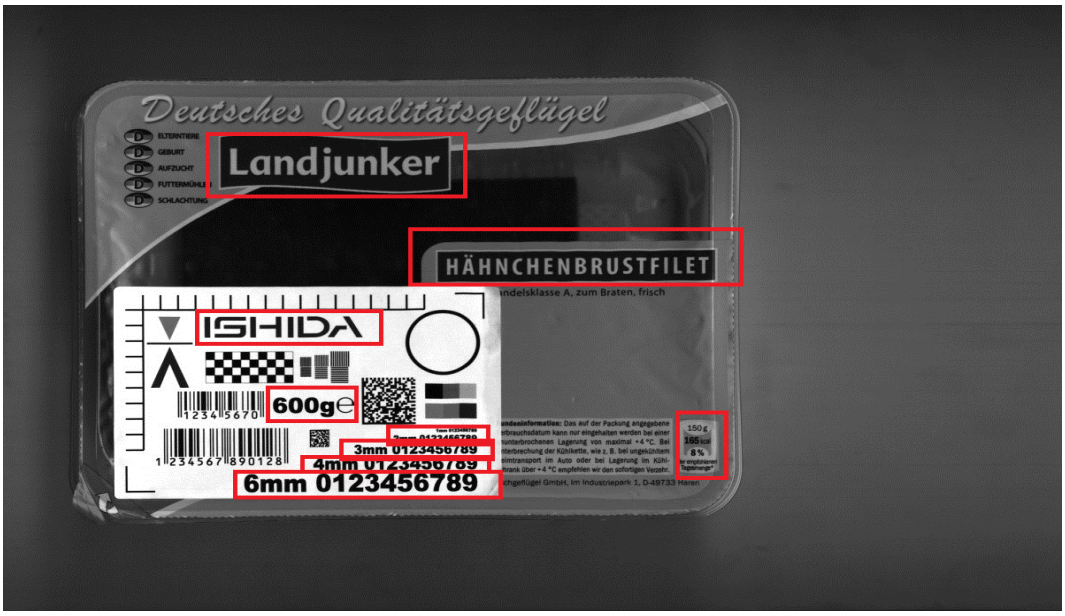


Figure 5.11: An ideal text localisation where text regions are marked. We need an automated procedure to detect these regions and create sub-images from them before OCR.

Figure 5.12: Text regions are detected using Neumann & Matas [2012].

Figure 5.13: Text regions are detected using Neumann & Matas [2012]. Comparing this example with Figure 5.12 shows that we need a robust text localisation algorithm that is not subject to small orientations.

Original Image

Output of the text localisation

Figure 5.14: Text regions are detected using Neumann & Matas [2012]. There are some false detections in the barcode region.

Figure 5.15: OCR results for the original image, filtered image and resized image for the output of Figure 5.14.

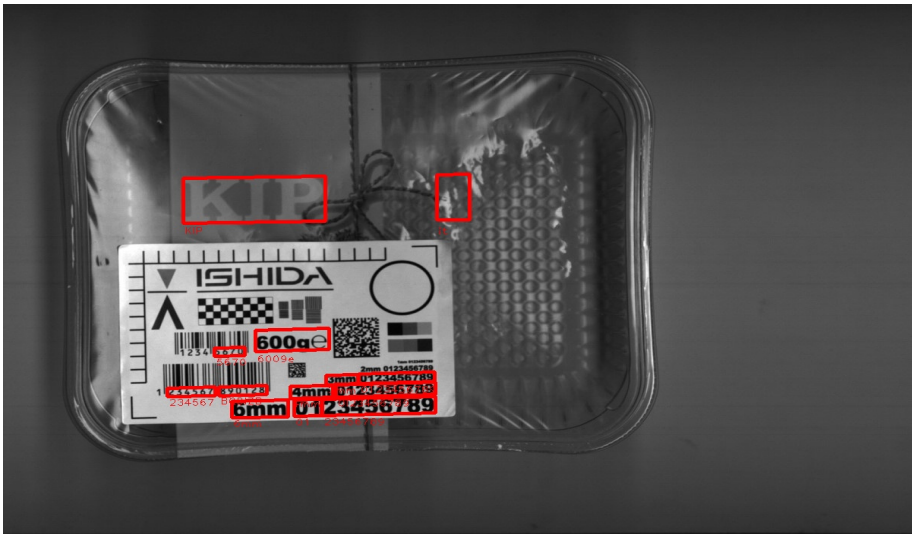


Figure 5.16: OCR results for a captured image applying web-based text localisation in http://www.textspotter.org/.

Figure 5.17: OCR results for a captured image applying web-based text localisation in http://www.textspotter.org/. The algorithm in http://www.textspotter.org/ shows robustness to orientation errors within a tolerance.

Chapter 6

Hossein Malekmohamadi TADD Technical Report for March 2014

6.1 Introduction

This is the March 2014 report for the project TADD. The core topic of this report is software development for QC. The job description for this month is based on Trainable Vision-Based Anomaly Detection & Diagnosis) Revised Second Level Project Plan. Amongst them, the focus is on: investigating high resolution camera, imaging set up, editing saved classifier, saving training sessions, fixing any outstanding bugs.

6.2 QA-TADD Development

In this section, first we give details on TADD software developments including tasks 1, 2, 5 and 8 introduced earlier. Moreover, we have investigated potential application of Halcon software for object identification and compare it with TADD capabilities.

6.2.1 TADD Software Developments

Finally, We are now able to compile both versions of TADD on a new machine. Here is the step by step installation guide:

1- The PC should be able to run CUDA v4.0+, where in this example We use CUDA 6. This is necessary for gslic algorithm.

2- In the project properties change the current CUDA 4.0 with CUDA 6.0 or any available version you have.

3- Right click and convert sub-projects starting with 'qt' to a QT-add in projects, e.g. qtcvoverlaywidget.

4- Change VC++ include, library,... directories accordingly.

5- In release mode add:
`C:/ProgramData/NVIDIA Corporation/CUDA Samples/v6.0common/inc` (only in Cuda/common/additional include)

6- In both UoL and SB PC there are Intel XE Studio and have been added to the project, we need to cut dependency on them. As a result Remove (ipps.lib, npp.lib, cuinj.lib) from linker inputs.

7. Add libiomp5md.dll and libmmd.dll to the release folder.

8. Beware of the architecture of the project. In the project properties window, select Configuration Properties — C++ — Code Generation. Select the "Enable Enhanced Instruction Set" line. This is to let compiler be aware of AMD64 processors if necessary.

9. Check qt and opencv libraries where they end in *d.lib and *d.dll.

10. If you want to change current opencv version to the latest, change library and include folders in the project. In case of compile error add: Add #undef min #undef max right before #include <gpu.hpp >.

11. We have two thresholding in image processing part of this project, one for potato counting and other one for background separation. The former can be changed in the source code and the latter is in the settings.ini file.

Another task for SW development that we carried out during March is both loading and saving classifiers are now working in release and debug mode. For this purpose, current loaded image and markup positions are saved in a directory and later they are called for loading. For editing, we are now able to see previously saved training session markups being loaded to the GUI for edit. User can now edit markups with left-click for adding new and right click for removing loaded markup locations.

Meanwhile with the help of a technician in UoL, a Gig-E camera has been tested to add to SB-TADD. The resolution of this camera is 2k×2k. We have written a new stand alone application to grab images from this camera and save it to hard disk instantly. This is integrated to SB-TADD where users can see the captured image. Currently, for classification purposes this image is resized to TADD native resolution, however We have to enable TADD to operate with high resolution images rather than resizing to 720p images.

There is an issue in the current software that stops further development and that is even reported by previous students. The bug is UI_webcamUI.h is not changeable. And after recompiling with new things it does not pick up the changes. It is under investigation.

Finally, the original SB-TADD consists of 16 sub-projects that has been pruned to 10 projects. This is done by removing each sub-project and test functionality of the project in both release and debug mode (See Figure 6.1).

6.2.2 Object Identification with HALCON

Halcon identify_vegetables.hdev demo have 14 object categories: apple, arugula, blueberry, bonbon tomato, carrot, champignon, cocktail tomatoes, corn salad,

cress, mangetout, potato, sweet potato, raspberry and strawberry Halcon identify_vegetables.hdev demo shows how to use the sample-based identification to identify different objects. First, the sample identifier is created, prepared, trained, and written to disk. In later runs, the trained sample identifier is read from disk to avoid time-consuming preparation step. In the run-time phase, the trained sample identifier is used to identify new images of vegetables. If the new trained sample identifier is not available yet it creates a new sample identifier. This demo uses the colour information in addition to the texture information for identification. Below are few remarks about this demo:
- Image size of 0.1 megapixel is enough for identification. - Halcon uses colour, texture and shape information. - Object names are prior information (preparation data). This is for supervised learning.

Differences to TADD are:

- TADD has the ability for users to interact and select a ROI and name the class (different blemish types) so an image as a whole can have multiple class names! - TADD is not using shape information of the objects however for later usage, e.g. 3D TADD, we can add this. - TADD is limited by minimum image resolution.

6.2.3 Conclusions

In March, main achievements are:
1. Installing both TADD softwares on a new machine and creating step by step guide in both release and debug modes.
2. Hardware setup of a new Gig-E camera and writing a new stand alone application and integrating it to SB-TADD.
3. Fixing load, save and edit classifiers in debug and release modes.
4. Removing unnecessary codes from SB-TADD and tested its functionality each time. The number of sub-projects are reduced to 10 from 17.

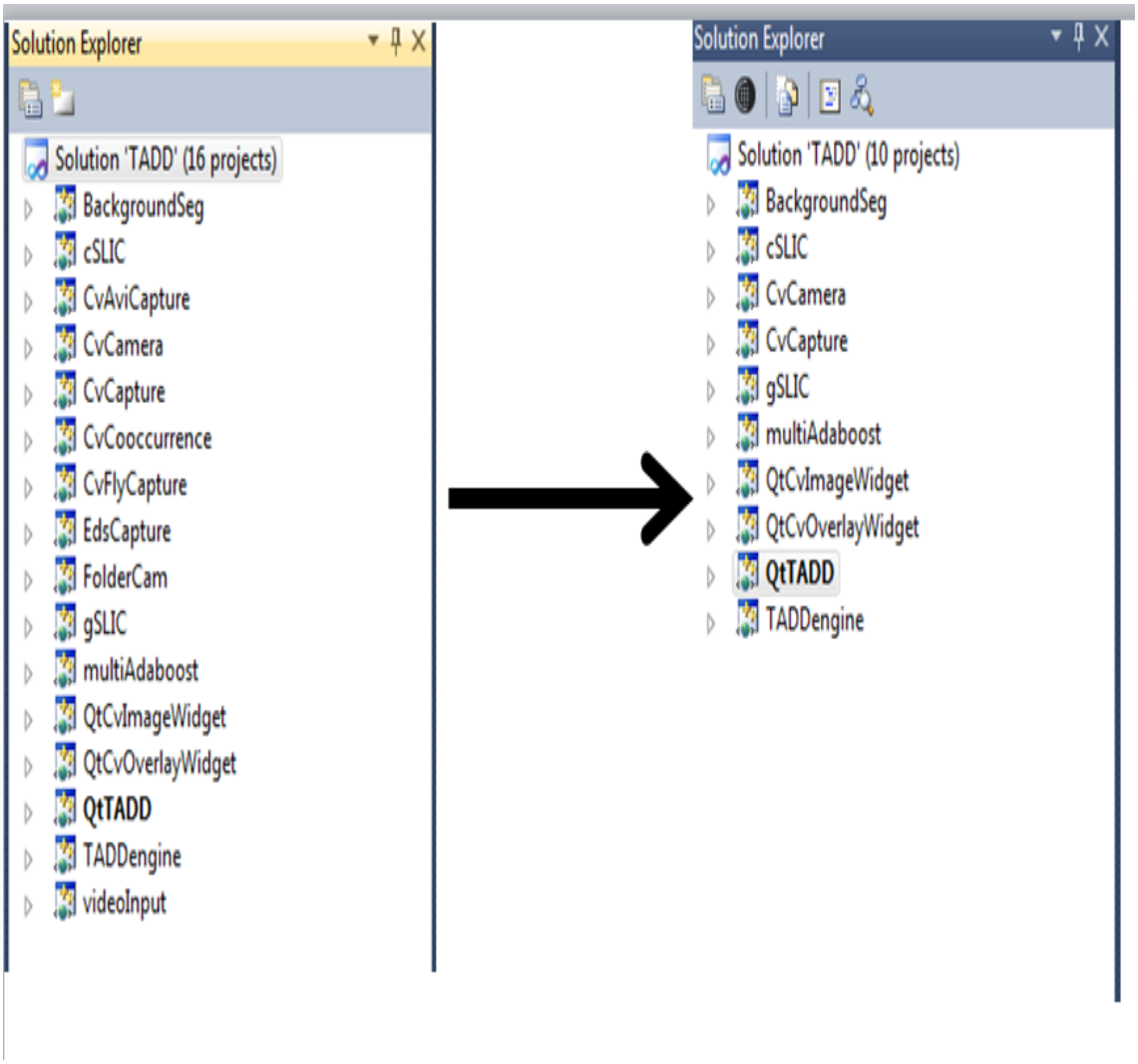


Figure 6.1: Pruning the project diagram

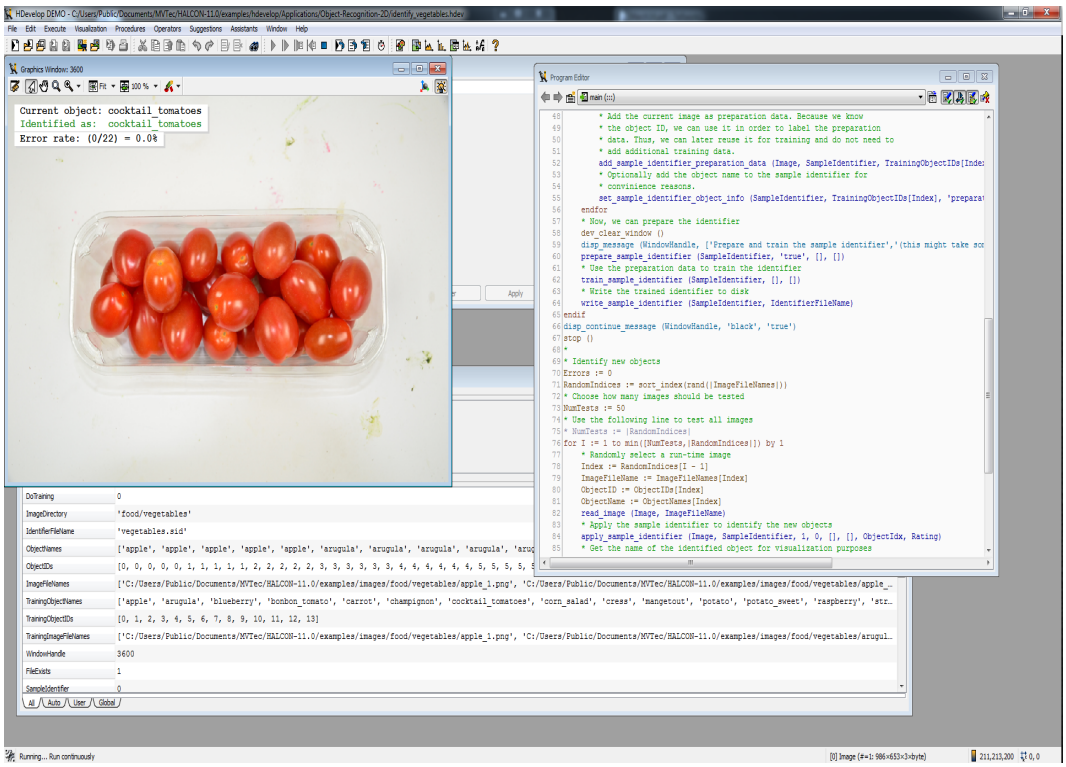


Figure 6.2: Example 1 of Halcon demo

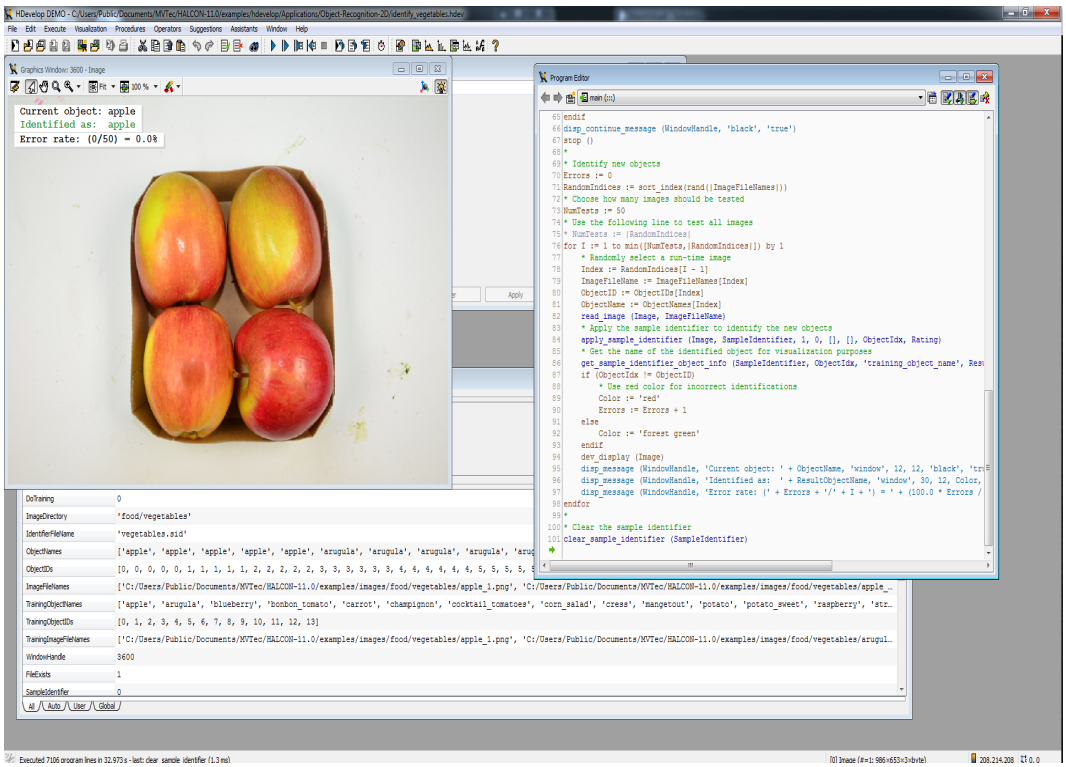


Figure 6.3: Example 2 of Halcon demo

Chapter 7

Hossein Malekmohamadi TADD Technical Report for April 2014

7.1 QA-TADD Development

The main QA TADD developments tasks in April 2014 include:

1- Adding the ability to edit saved classifiers to TADD v1 and v2– Both softwares are now able to be fed with saved classifier and user can edit the training data to improve the results.
2- Adding Pulnix camera to TADD-v2. The software is now implemented to TADD v2 where it can save the captured image on disk and this image is called instantly inside TADD-v2.
3- Created Universal TADD with classification potentials from TADD v1 and user friendliness of TADD v2. As an example, it has zoom and log functions from TADD v2 but the classification method is based on TADD v1.
4- Last but not the least, TADD team has submitted a paper to BMVC based on QA TADD. We compared the results to a state of the art system. Please see the attachment.

References

Bay, H., Tuytelaars, T. & Van Gool, L. (2006). Surf: Speeded up robust features. In *Proc. ECCV*, 404–417, Springer. 6, 15

Neumann, L. & Matas, J. (2012). Real-time scene text localization and recognition. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 3538–3545, IEEE. v, 26, 34, 35

Tighe, J. & Lazebnik, S. (2010). Superparsing: scalable nonparametric image parsing with superpixels. In *Computer Vision–ECCV 2010*, 352–365, Springer. 25

Semantic Analysis of Food Images via Interactive Training and Classification

BMVC 2014 Submission # 237

Abstract

We present a system for the semantic analysis of food images. It is a feasible solution to a variety of real quality control tasks mainly because it adopts a novel interactive training and classification mode. In general, our system incorporates the interactive acquisition of training data into the automatic classification based on adaptive boosting. To reduce the manual effort required for interactively acquiring the online training data, the input images are efficiently processed through GPU-based over-segmentation. We also developed a user-friendly graphical user interface to allow users to edit (select, deselect and reselect) the training data. It also allows the users to add more training data to improve the classifier if necessary. The experiments demonstrate that our system efficiently delivers a semantic analysis for food images. More importantly, different from the state-of-the-art methods which typically rely on offline training using a large collection of data, our system can rely only on a tiny amount of training data offered by the users in an interactive manner and achieve a highly accurate classification.

1 Introduction

It has been more than two decades since computer vision techniques were applied to automatic inspection of food products for quality assurance [6]. The essential motivation is to use such techniques to replace human inspectors for the evaluation of a variety of quality attributes of raw and prepared foods [7]. Benefitting from the rapid development of commercial computing and sensing technologies, computer vision systems designed for food quality assurance have been able to provide a high level of flexibility and repeatability at relatively low cost. In sharp contrast, labour cost has been widely increased in the food industry where manufactures typically have a low profit margin. Also, consumers tend to buy the food products which look good since in most cases, it is not possible to directly taste the food when buying it. Hence, many companies in the food industry are willing to invest in proper computer vision techniques to reduce the cost as well as to boost sales.

The core task of vision-based food quality assurance is visual anomaly/defect detection, usually followed by diagnosis/grading. Existing techniques involve various image analysis methods mostly in the fields of feature extraction, semantic segmentation and classification.

1.1 Existing techniques

An intelligent vision system for detecting defects on chicken meat was presented in [3]. Defects were detected based on the analysis of the chromatic content of chicken images.

© 2014. The copyright of this document resides with its authors.
It may be distributed unchanged freely in print or electronic forms.

Possible defect areas were first extracted by means of morphological image reconstruction and then classified according to a predefined list of defects. A system was presented in [15] for monitoring the production of fast food using a single static camera. It analysed the video through some low level techniques including skin detection, shadow detection and region growing and then produced a description of the fast food. This description gave the ingredients used in the food and their physical arrangement to make sure that the food was produced correctly. Colour information was used in [9] for monitoring food processing. The proposed system detected features based on colours and then employed PCA to characterise target features in colour space from a set of training data, and was easy to train, efficient and effective. Colour features were also investigated in [8] where the food image recognition system also used other types of features such as texture and SIFT [10]. Then, the multiple kernel learning method was employed to estimate the optimal weights to combine these features for each category of food. Another food recognition system presented in [18] utilised a novel representation for food items that calculates pairwise statistics between local features computed over a soft pixel-level segmentation of the image into eight ingredient types. The following discriminative classification was based on the multi-dimensional histogram of these statistics. The method proposed in [4] detected irregularities for fruit inspection by comparing the test image to a reference image representing the 'good' case (expected/normal patterns). It generalised from the references and composed the test images to find regions with low composition likelihood, which were considered as defects. To detect multiple food items from one food image, a method was proposed in [11] based on co-occurrence statistics and manifold ranking. In [6], the target was automatic identification and quantity estimation for Chinese food. The identification task was solved as a classification problem where multi-label SVM classifiers were trained for the features including SIFT with sparse coding, LBP [12] with multi-resolution sparse coding, colour histograms and Gabor texture features. Then, the quantity of the identified food was estimated based on exploiting the depth information.

1.2 Our system

It can be seen that there are several common issues existing in the vision-based techniques for food processing. Firstly, the efficiency of the technique is vital since food products are typically fast-moving consumer goods. It is common that a huge amount of food products need to be handled every day in a food company.

Secondly, due to the variety and the variation of food, most of the systems require a large dataset for training purpose in order to cover each type of defects which could be extremely occasional, but does exists. However, it is not always easy to collect sufficient data for training because a food product could change rapidly in terms of colour, intensity, shape and volume, and these features are important in most of the aforementioned techniques. Typically, such a data collection could take a couple of months (e.g., the Supermarket Produce dataset was the result of 5 months of on-site collecting in a local fruits and vegetables distribution centre [14]).

Thirdly, since usually a food company does not only process one type of food products (especially if the food is seasonal), the systems need to be periodically retrained to adapt to different food products. This means that the time-consuming offline training, as well as the extra effort needed to prepare the new training data (e.g., manually labelling the images of another product) could be very inconvenient for practical applications.

Finally and most importantly, to the best of our knowledge, there is no manual 'correction

mechanism' involved in any of the existing vision-based systems for food quality assurance. This is a waste of the expertise of the very people using the systems. They are typically quality control inspectors who possess a significant amount of knowledge about the possible defects. Moreover, safety standards on food products are quite strict since a bad product could cause serious troubles and the producer might face huge fines and compensations. Therefore, from a practical point of view, it is very important to introduce some manual correction mechanism as there is no guarantee that the automatic system can always deliver successful quality inspection. When the automatic system cannot recognise a certain defect, the human inspector should be allowed to step in and offer the particular help to the system to correct the error.

These four issues form the motivation of developing our novel system. Its most novel component is the idea of using interactive training to improve the classification results in real-time. To realise this idea, we develop a novel mechanism to help users to conveniently select the training data that they think necessary in an interactive manner. An adaptive boosting (AdaBoost) classifier [16] is then trained using the manually selected data and the test image can be assessed by the trained classifier. In the next section, we give the technical details of the user-friendly mechanism which enables the interactive training, as well as the classification based on a set of image features selected using AdaBoost.

2 Classification via interactive training

Normally, to train the classifiers and test their performance, the images are marked up by hand to provide the ground truth information indicating the correct class of each pixel. Most existing publicly available databases which benchmark image segmentation techniques work in this way. However, due to the variety of food, it is not realistic to apply such a strategy to the real tasks of food quality inspection. Note that one specificity of food is that its appearance changes rapidly and significantly over time and in many cases, time is the very reason that some defects occur. This is quite different from the objects in the image databases (e.g., the MSRC image database[1]) for benchmarking general image segmentation or object classification methods where the objects are generally invariant within a certain period of time. Such a specificity makes the conventional offline training based on a large set of previously constructed ground truth data not reliable. This is why it is so important to implement the online training using the present data. To achieve this, we developed the system illustrated in Figure. 1.

In Figure. 1, the training mode is designed for iteratively teaching the adaptive boosting algorithm to distinguish between different categories. This stage requires the user to select and label a set of anomalies, which is used later for training the system. The user is then able to receive feedback by visually inspecting the accuracy of the classification. If mistakes are being made the user simply retrains the system. Once trained the system can be used in deployment mode, which means no more training is required to classify and diagnose anomalies. The user is able to swap back to training mode at any time to retrain after any classification mistakes. This procedure means there is no limitation on how many times the system can be trained, relying on experts to teach and guide the machine learning algorithm.

[1] http://research.microsoft.com/en-us/projects/objectclassrecognition/

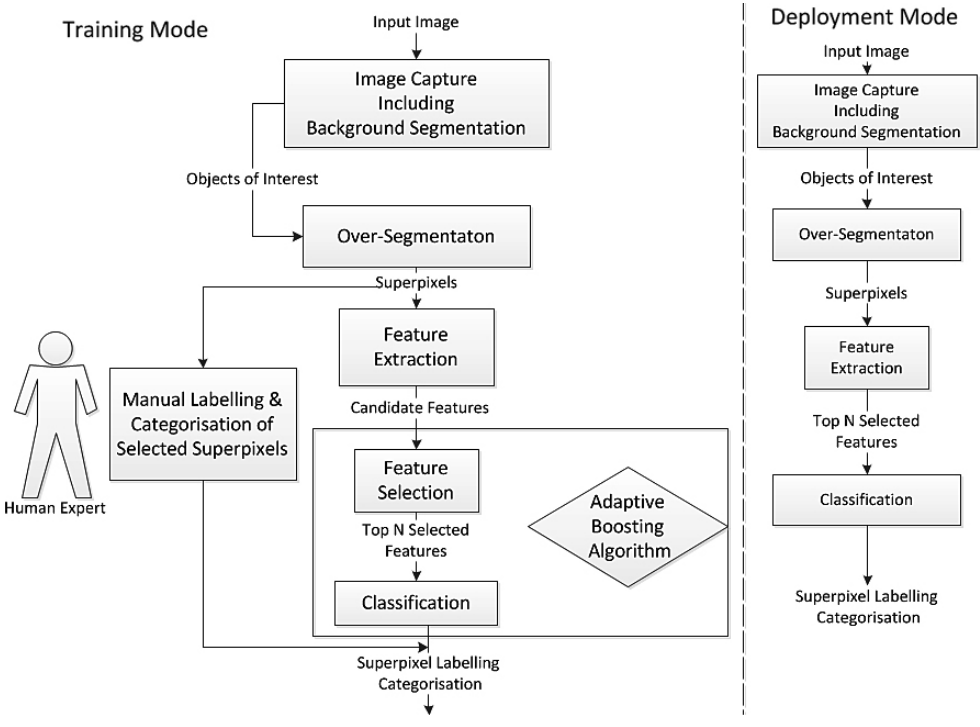


Figure 1: Overview of our system

2.1 Background segmentation

The proposed system use a simple thresholding method for background segmentation. After the background segmentation, the user is not allowed to select any region in the background as the training data.

2.2 Over-segmentation

Image over-segmentation is used in our system to enhance the interactive mark-up. The process of segmenting the captured image into clusters (i.e., superpixels) which follow the contour of edges by colour differences aims to reduce redundancy in the image. Superpixels also allow an expert to select a relatively large area with only a single mouse click, which significantly mitigates the manual effort needed in the interactive scheme and makes it much more efficient. We use the SLIC algorithm [1] for over-segmentation where to improve the efficiency, we apply the GPU version of SLIC [13] in our system. It works by loading the cluster's information into local shared memory. In each iteration, after all pixels have been assigned a label, one thread per cluster is used to update the centres.

2.3 Feature extraction

In our system, a large collection of candidate features are required for each superpixel, from which the AdaBoost algorithm can select to build the classifiers. As suggested by [2], we use

7 colour channels: raw RGB, normalised RGB and the intensity channel. In [2], the image regions used for feature extraction were squares of size $33 \times 33, 65 \times 65, 97 \times 97, 129 \times 129$ and 161×161 pixels. Our system differs by using only superpixel regions, classifying each cluster of pixels based on their region characteristics. Superpixels reduce computation requirements in two aspects. Firstly, an image sized 1280×720 is represented as around 9200 clusters requiring only 1% of the computational resources when compared to pixel-wise classification. Secondly, as the superpixel regions adapt to cluster together similar pixels, compensating for the requirement of using multiple region sizes.

We then consider three statistical moments for each superpixel: average (Eq. (1)), variance (Eq. (2)) and skewness (Eq. (3)).

$$\bar{x} = \frac{1}{N} \sum_{N}^{j=1} x_j; \tag{1}$$

$$Var(x_1 \ldots x_N) = \frac{1}{N-1} \sum_{N}^{j=1} (x_j - \bar{x}); \tag{2}$$

$$Skew(x_1 \ldots x_N) = \frac{1}{N} \sum_{N}^{j=1} \left(\frac{x_j - \bar{x}}{\sqrt{Var(x_1 \ldots x_N)}}\right)^3. \tag{3}$$

The first statistical moment is difficult to parallelise as this requires the implementation of a prefix-sum for the most optimal implementation, adding complexity with the use of shared memory. We use the large amount of segments produced by the gSLIC algorithm and sum each region on a separate thread using cluster centres. Once the first statistical movement has been calculated both variance and skewness are simple to implement.

We calculate the three moments in each channel, resulting in 21 features. We also use a Sobel edge detector and a range filter in the intensity channel respectively. The aim of the edge detector is to determine the rate change in pixel intensities, which is a useful candidate feature for defects with a high rate of change. The range filter calculates the minimum, maximum and difference between the maximum and minimum values within a 5×5 neighbourhood. It indicates the roughness of a texture, aiding in the identification of defects. Then we compute the 3 moments on each of the resultant maps, which produces 6 more features. We also record the longest Sobel edge length in each superpixel as a feature, which has been demonstrated experimentally useful for improving classification of some defects. Therefore, we generate 28 features in total.

2.4 Classification via AdaBoost

We employ the Adaptive Boosting (AdaBoost) algorithm proposed in [2] to automatically select the best features for discriminating between defects and normal regions. The reason that we apply this algorithm in our system is that it can handle the natural variation in fresh produce due to different seasons and lighting conditions [2]. Also, the method is able to build 'minimalist' classifiers that optimise detection performance at low computational cost, which is very important for a real-time interactive system.

This approach involves two stages: the first stage selects a feature set that will be used to train an AdaBoost classifier in the second stage. It also includes an additional step in order to limit the number of unique features used in the final classifier to a smaller number than the total number of weak classifiers allowed.

With these techniques combined, our system allows users to select a handful of superpixels as the training data and the system can show the classification results based on the AdaBoost classifier trained by them in less than 1 second including visualisation. The users can then interactively improve the results by selecting new superpixels (or deselecting existing superpixels). We also submit a video as our supplemental material to show how our system works.

3 Experimental results and analysis

This section describes two methods of evaluating the prototype system. The first method focuses on interactivity of this system for industry applications whilst the second method shows the classification accuracy with respect to ground truth data for training and testing.

3.1 Interactive Training

Interactive training and testing of the prototype system involves loading an image, creating editable mark ups, labelling classes, training and testing the classifier. Images used in the assessment of this system are captured from a digital camera with the resolution of 1280×720. Users can label classes, e.g. Good, Blemish,..., and mark up superpixel locations for each class. After pressing the training button of the GUI, the system learns. For the testing phase, the user can assess the currently marked up image to have it fully assessed or load a new image and test it. Other available options for users are to save, load and edit classifiers. The advantage of the prototype system is its speed and accuracy. Speediness comes from the GPU-based superpixel segmentation and GPU-based implementation of the learning algorithm. Accuracy is based on the feature extraction and strong classifier algorithm that is based on mixing several weak learners. For the first advantage (speed) please refer to the uploaded supplementary material. And for accuracy, the next subsection contains details of the further experiments.

3.2 Ground Truth Training and Testing

In the experiments, two datasets were used. One is a collection of images of 25 white potatoes and the other is that of 41 red potatoes. In each dataset, we only use the ground truth images of 5 potatoes for training. The reason that we only select a small number of images for training is that in real applications, users often tend to label a small amount of regions, which requires that the classifier must work based on a small amount of training data.

Some of the results are shown in Fig. 2 in association with the corresponding original images and the pixel-wise ground truth images marked by an expert in potato pathology. Note that some areas of high uncertainty or ambiguity were left unmarked, which is inevitable since even under a microscope, the expert could still be unsure about whether a particular region is good or not. These unmarked pixels are ignored during the training of the classifier. Also, our system relies on superpixel-based over-segmentation for the reasons of efficiency and convenient selection of training data. Some holes thus occur, particularly in the regions where the labels change as shown in Fig. 2 because one superpixel could contain pixels with different labels.

To compare the results of the discussed prototype, the state-of-the-art method introduced in [17] is used for training and testing with the same potato dataset. The system allows the

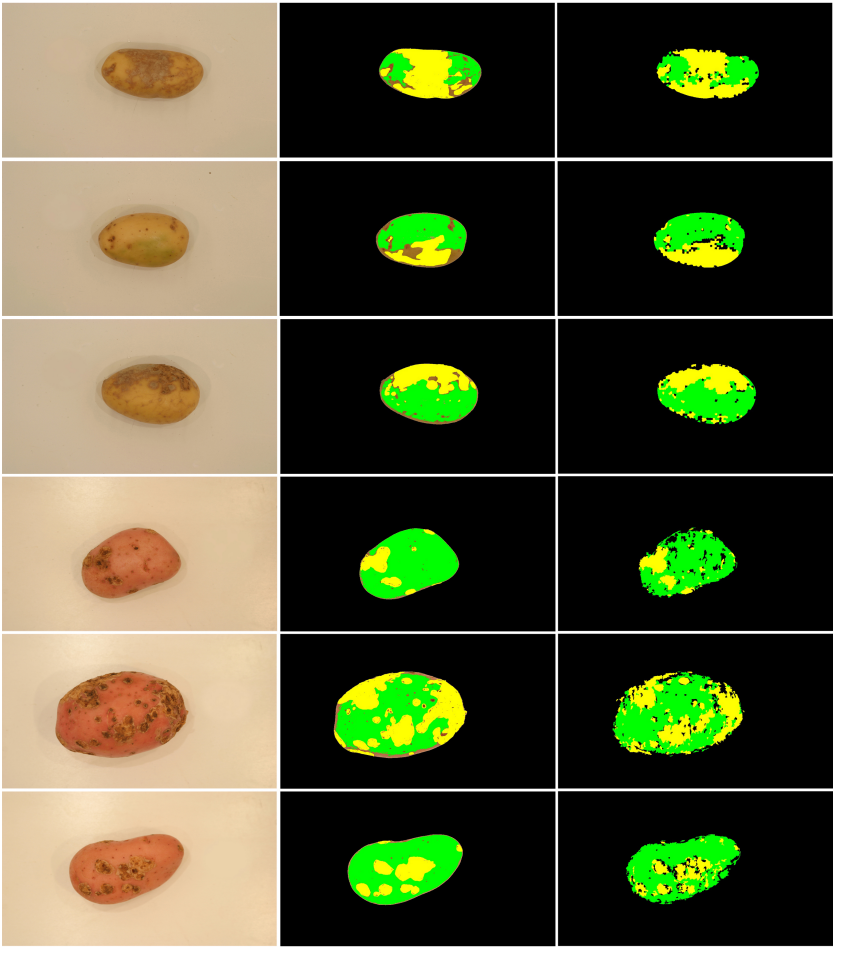


Figure 2: Some of the results produced by the proposed system where we use green and yellow to denote good regions and blemishes respectively. Left column: original images; Middle column: ground truth images; Right column: our results.

user to perform segmentation and classification. Users need to mark up locations and provide training data pixel by pixel. This is typically a very tedious task. Classification is based on random forests. This system utilises up to three spatial and one spectral dimension in the feature calculation. It has capabilities like saving classifiers for later batch processing.

To perform the evaluation, we use the ground truth images of 5 potatoes in each dataset for training. This is actually impossible for the method in [17] in practice because human users cannot mark up 5 images with a resolution of 1280×720 pixel by pixel (but possible for our system based on superpixel which significantly mitigates the manual effort of mark-up). Even so, comparison results of the system in [17] to our system is shown in Figs. 3 and 4.

In these figures, we use the following equations to compute the accuracy in detecting the

Figure 3: Results for 25 white potatoes trained and tested with the proposed system and the system in [□]

good and the bad areas respectively.

$$Acc_{good} = \frac{|G_r \cap G_g|}{|G_r \cup G_g|}, \quad Acc_{bad} = \frac{|B_r \cap B_g|}{|B_r \cup B_g|} \tag{4}$$

where G_g and B_g denote the good and the bad areas (calculated by the number of pixels) in the ground truth images while G_r and B_r denote the good and the bad areas in the resultant images.

On average, the accuracy of the system in [□] in detecting the good areas for the white potatoes is 0.8807 and for the bad areas is 0.8455, while the average accuracy in detecting the good areas for the red potatoes is 0.8785 and for the bad areas is 0.8801. In comparison, the average accuracy of our system in detecting the good areas for the white potatoes is 0.8451 and for the bad areas it is 0.7983, while the average accuracy in detecting good areas for the red potatoes is 0.9372 and for the bad areas it is 0.9099. It can be seen that in terms of accuracy, our system is comparable with the state-of-the-art method. The results of the prototype system introduced in this paper, especially in detecting bad areas, can be improved by using some other features like frequency domain features. However, the prototype system provides a trade-off between accuracy and speed that makes it suitable for training in quality control tasks. Also, note that our method is based on labelling superpixel regions while in one superpixel there could be different types of potato regions (good or bad) but the competing method is pixel-wise. When we use the ground truth images of 5 potatoes to train our system, we always need to first convert the pixel-wise ground truth images into superpixel-wise ones. Such a conversion inevitably introduces errors. Furthermore, it is worth pointing out that in order to facilitate the evaluation, all of the experiments were implemented based on a 'one-off' operation, which means that we actually disable the powerful function of the system that

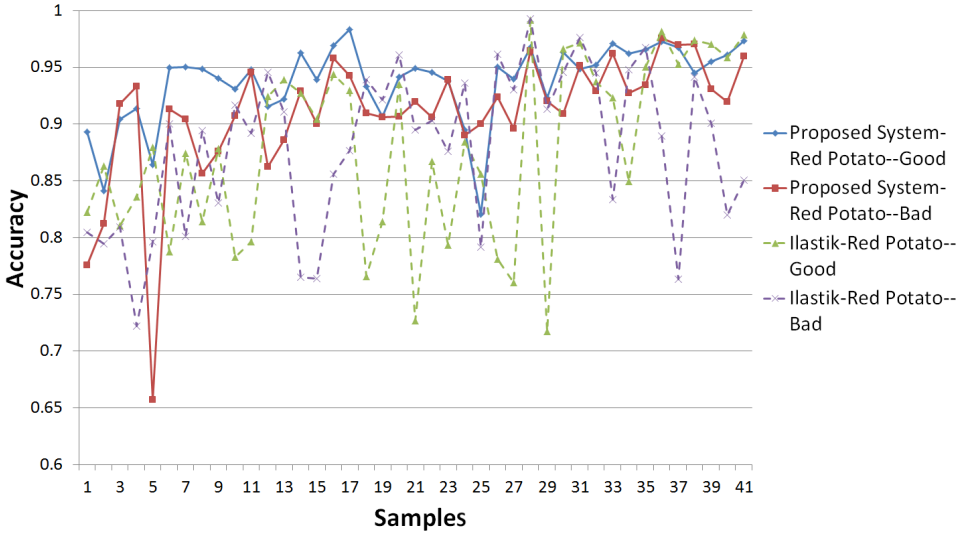


Figure 4: Results for 41 red potatoes trained and tested with the proposed system and the system in [17]

allows the user to modify the training data interactively. In general, the segmentation output by our method can be easily improved by interactively modifying the training data. This can be demonstrated by the supplemental video.

4 Speed

Obviously the speed of the proposed system will vary, mainly depending on how many training data the user selects. And usually, the more selected training data, the more accurate the classification results. We thus demonstrate that our system can be implemented in real-time through the video submitted as the supplemental material. The video recorded typical operations of the system using a PC with a dual core 2.13GHz CPU, 4GB RAM and a GeForce GTX 570 graphics card.

5 Conclusions

We propose a prototype system for the semantic analysis of food images via interactive training and classification. The entire idea, an integration of background segmentation, superpixel over-segmentation, a selection of specific features and the AdaBoost classification is promising according to our preliminary evaluation and comparison. The core contribution is the novel interactive training and classification framework which gives the system advantages in various aspects.

Future work will focus on investigating how to use 3D information for more accurate analysis. For example, the presented system does not consider the 3D shape of food, resulting in a defect seeming smaller toward the edge of the food since it is at a steeper angle to the camera. This challenge could be approached by using the 3D depth information.

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2281, 2012.

[2] Michael Barnes, Tom Duckett, Grzegorz Cielniak, Graeme Stroud, and Glyn Harper. Visual detection of blemishes in potatoes using minimalist boosted classifiers. *Journal of food engineering*, 98(3):339–346, 2010.

[3] Mauro Barni, Vito Cappellini, and Alessandro Mecocci. Colour-based detection of defects on chicken meat. *Image and Vision Computing*, 15(7):549–556, 1997.

[4] Oren Boiman and Michal Irani. Detecting irregularities in images and in video. *International Journal of Computer Vision*, 74(1):17–31, 2007.

[5] Mei-Yun Chen, Yung-Hsiang Yang, Chia-Ju Ho, Shih-Han Wang, Shane-Ming Liu, Eugene Chang, Che-Hua Yeh, and Ming Ouhyoung. Automatic chinese food identification and quantity estimation. In *ACM SIGGRAPH Asia Technical Briefs*, 2012.

[6] Davies E.R. *Machine Vision*. Academic Prss Limited, London, 1990.

[7] Sundaram Gunasekaran. Computer vision technology for food quality assurance. *Trends in Food Science & Technology*, 7(8):245–256, 1996.

[8] Taichi Joutou and Keiji Yanai. A food image recognition system with multiple kernel learning. In *Proceedings of International Conference on Image Processing*, pages 285–288, 2009.

[9] Kok-Meng Lee, Qiang Li, and Wayne Daley. Effects of classification methods on color-based feature detection with food processing applications. *IEEE Transactions on Automation Science and Engineering*, 4(1):40–51, 2007.

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

[11] Yuji Matsuda and Keiji Yanai. Multiple-food recognition considering co-occurrence employing manifold ranking. In *Proceedings of International Conference on Pattern Recognition*, pages 2017–2020, 2012.

[12] Timo Ojala, Matti Pietikainen, and Topi Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002.

[13] Carl Yuheng Ren and Ian Reid. gSLIC: a real-time implementation of slic superpixel segmentation. Technical report, University of Oxford, Department of Engineering, 2011.

[14] Anderson Rocha, Daniel C Hauagge, Jacques Wainer, and Siome Goldenstein. Automatic fruit and vegetable classification from images. *Computers and electronics in agriculture*, 70(1):96–104, 2010.

[15] Richard Russo, Mubarak Shah, and Niels Lobo. A computer vision system for monitoring production of fast food. In *Proceedings of Asian Conference on Computer Vision*, 2002.

[16] Robert E Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine learning*, 37(3):297–336, 1999.

[17] Christoph Sommer, Christoph Straehle, Ullrich Kothe, and Fred A Hamprecht. ilastik: Interactive learning and segmentation toolkit. In *Proceedings of IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 230–233, 2011.

[18] Shulin Yang, Mei Chen, Dean Pomerleau, and Rahul Sukthankar. Food recognition using statistics of pairwise local features. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 2249–2256, 2010.