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

October 2014

Abstract

This technical quarterly report is a summary of the work we have done in August, September and October 2014 for the project Trainable Vision-based Anomaly Detection and Diagnosis (TADD). We just integrate the corresponding six monthly technical reports produced by Dr Ran Song and Dr Hossein Malemohamadi respectively.

TSB-funded Project 'TADD' –
Trainable vision-based anomaly detection and diagnosis
Technical Report for August 2014

Ran Song and Tom Duckett

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

Abstract

Since the last project review meeting in 9 August 2014, I have spent most of my time in August in preparing the Siggraph presentation, attending the Siggraph conference and travelling abroad for holiday. Thus in the following, I just attach the report on Siggraph 2014.

Report on Siggraph 2014

Ran Song

I attended the renowned Siggraph conference in Vancouver, Canada from 10 August to 14 August, 2014. It is the most prestigious forum for the publication of computer graphics research. This year, 14,045 attendees including research scientists, developers, filmmakers, students and academics from 50 U.S. states and 75 countries gathered to experience cutting-edge computer graphics and interactive techniques. It was heavily covered by media and quite a lot of major IT companies demonstrated their latest techniques and/or research findings at Siggraph 2014, for example:

Microsoft: http://research.microsoft.com/en-us/about/siggraph-2014.aspx
Intel: https://software.intel.com/en-us/siggraph2014/home
NVIDIA: http://www.nvidia.com/object/siggraph2014.html

Figure 1. The exhibition hall © Siggraph 2014

I presented a technical paper titled ‘Mesh saliency via spectral processing’ at the conference and also talked with people from industry. There were quite a few 3D sensors on display in the huge 40,000 sq. ft. exhibition hall (see Figure 1). Most of them were designed for 3D motion capture (for the applications in film industry for special visual effect and video/computer games) while several of them were apparently fit for sensing small static objects with high accuracy. It seems that the stereo vision-based sensing techniques are quite popular and demonstrated to be reliable, efficient and partially noise-free.

Besides the hardware, I also found some interesting software which can be potentially used in the TADD project. Certainly most of them are not free but some are indeed open source. One is called ‘Open Standard APIs for Vision and Camera Processing’ developed by the KHRONOS Group. It includes three libraries, namely OpenKCam for Camera Processing, OpenVX for powerful efficient vision acceleration and EGL stream I/O. As shown in Figure 1, it is complementary to OpenCV, which means our TADD software based on OpenCV could be updated with the help of OpenVX for acceleration purpose.

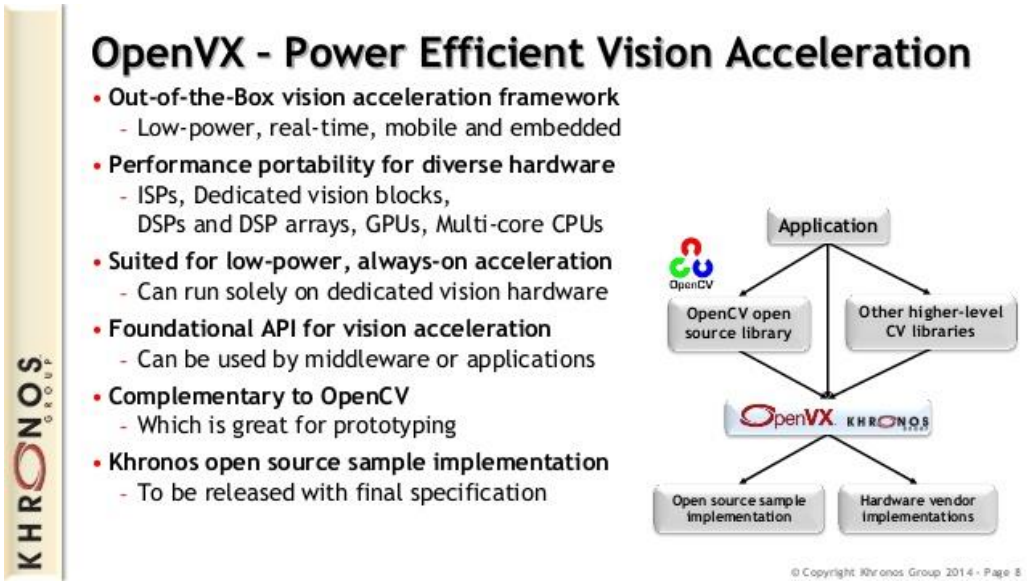


Figure 2. An overview of the the OpenVX API. © Copyright Khronos Group

Most of the software for vision applications, free or not free, support OpenCV, which demonstrates the fact that OpenCV has been the dominant industrial standard. However, for 3D graphics applications, there are three major players: OpenGL, CGAL (https://www.cgal.org/) and DirectX. Compared with OpenGL and DirectX, CGAL is not that well-known. However, their staff at the booth gave me a very impressive demonstration about what it can achieve for 3D geometry computation, rendering and visualisation. Obviously, the development of the 3D-TADD software could benefit from CGAL.

In summary, attending Siggraph 2014 is an excellent experience for me to know about the cutting-edge R&D in 3D vision and graphics around the world. More importantly, I received valuable advices about how to make the research work come out of the lab and become really useful for industry to solve real-world problems. I believe that it is particularly vital for the success of the TADD project.

TSB-funded Project 'TADD' - Trainable vision-based anomaly detection and diagnosis Technical Report for September 2014

Ran Song and Tom Duckett

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

Abstract

In September 2014, I worked for both Label-TADD and QA-TADD. Thus this report is composed of two parts. The work related to the Label-TADD includes software debugging and efficiency test. The work about the QA-TADD is the computation of the confusion matrix used for evaluating the performance in the tasks of multi-label classification.

1. Label-TADD debugging and updating

Occasionally, we found a bug within the original Label-TADD software. According to its specification, once the training has been done, it should be able to handle an indefinite number of test images where label region is detected in each test and OCR is also implemented. However, we found that after processing 6–8 images, the software would suffer from a crash. We manage to pinpoint the bug. It is caused by an accumulated change added to two vital parameters used as thresholds which control the number of corresponding SURF feature points (matching points). After 6–8 runs, these threshold become very intolerant and tend to reject all of the candidate matches. Consequently, there is no input data available for a built-in function of OpenCV which computes the homography (the transform matrix describing the motion between two images) based on the matching points. However, in terms of computer vision textbook, the computation of the homography requires at least four pairs of noncolinear matching points. As a result, when we call this function in our codes, the software crashes and no reason is given. This

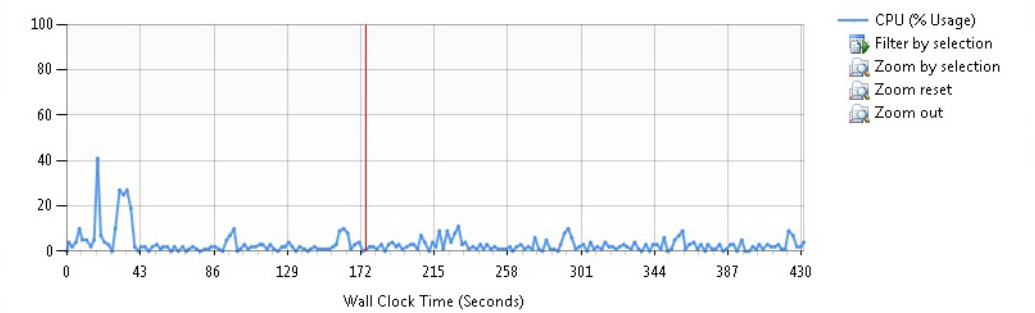
```
1047    img_scene.release();
1048    H.release();
1049    descriptors_scene.release();
1050    vector<Point2f>().swap(scene);
1051    vector<Point2f>().swap(obj);
1052    vector<KeyPoint>().swap(keypoints_scene);
1053    vector<DMatch>().swap(matches);
1054    vector<DMatch>().swap(good_matches);
1055    scene.clear();
1056    obj.clear();
1057    keypoints_scene.clear();
1058    matches.clear();
1059    good_matches.clear();
1060    matcher.clear();
```

Figure 1: A piece of codes for solving memory leak

is usual for open source libraries such OpenCV where the input check for its built-in functions sometimes is missing. Once we pinpoint the bug and find out how it is generated, the solution is simple. In the updated codes, we always reset these two parameters to default values at the beginning of every run.

Also, we found that the original software would slow down a little bit (by 30% roughly) after processing a large number of images. We also managed to sort this problem. It is mainly caused by memory leak, which makes the accessible memory decreased. In other words, although my desktop computer has 16GB installed memory (RAM), after keeping running for a while, it is equivalent that the software is being implemented on a computer with 11GB installed memory. Fig. 1 shows a piece of the newly added codes for avoiding memory leak. Fig. 2 shows the efficiency analysis of the updated version of the software. The running time of a specific function is proportional to the percentage of exclusive samples. It can be seen that there has been little room for the current program to further accelerate since the most time-consuming functions are OpenCV built-in functions which we assume have been optimised by its supporting team.

We have also done some initial analysis for the performance of the current software which now typically spends less than a second to process one test image with a resolution of 1197×777. We also suggest that after closing the GUI for exiting the software, it is better to make a double check through task



Hot Path

The most expensive call path based on sample counts

Function Name	Inclusive Samples %	Exclusive Samples %
TADD.exe	100.00	0.00
Concurrency::details::ThreadProxy::ThreadProxyMain(void *)	**72.92**	**0.00**
_WinMainCRTStartup	**26.59**	**0.00**

Related Views: Call Tree Functions

Functions Doing Most Individual Work

Functions with the most exclusive samples taken

Name	Exclusive Samples %
cv::Mat::at<unsigned char>(int,int)const	17.84
cv::SURFInvoker::operator()(class cv::Range const &)const	13.34
cv::calcHaarPattern(int const *,struct cv::SurfHF const *,int)	8.37
cvFloor(double)	7.34
cvRound(double)	6.25

Figure 2: Efficiency analysis of the program where the running time of a function is proportional to its corresponding exclusive samples.

manager and terminate 'TADD.EXE' in the 'Processes' list to completely release the memory if it exists.

2. QA-TADD Evaluation

In September, we also implemented quantitative evaluation for the QA-TADD system. We computed the confusion matrices in order to evaluate the performance of the QA-TADD in the tasks of multi-label classification. In the field of machine learning, a confusion matrix, also known as a contingency table or an error matrix, is a specific table layout that allows visualisation of the performance of an algorithm, typically a supervised learning one. Each column of the matrix represents the instances in a predicted class, while

each row represents the instances in an actual class. The name stems from the fact that it makes it easy to see if the system is confusing two classes. Confusion matrix is also a widely used statistic in computer vision to evaluate the accuracy of the classification. More importantly, it tells us what really happens when something goes wrong, which usually helps to make potential improvement aiming at the shortcomings of the current method.

We computed confusion matrices for a set of 41 images which contain different types of potato defects such as scab, greening, black dot and other types of blemishes. Figs. 3–8 show the resultant confusion matrices of 6 input images. For example, to understand confusion matrix, in Fig. 3, 21596 pixels detected as 'Scab' are truly 'Scab' pixels while 869 pixels detected as 'scab' are actually 'Unblemished' pixels. Also in Fig. 3, there should be no 'Greening' pixels but 104 'Scab' pixels are incorrectly classified as 'Greening' pixels. This is consistent with the classification results and the ground truth shown in Fig. 3.

It can be seen that typically, there are quite a lot of pixels correctly labelled as 'Unblemished'. This suggests that the system should have an excellent performance in the tasks of two-label classification.

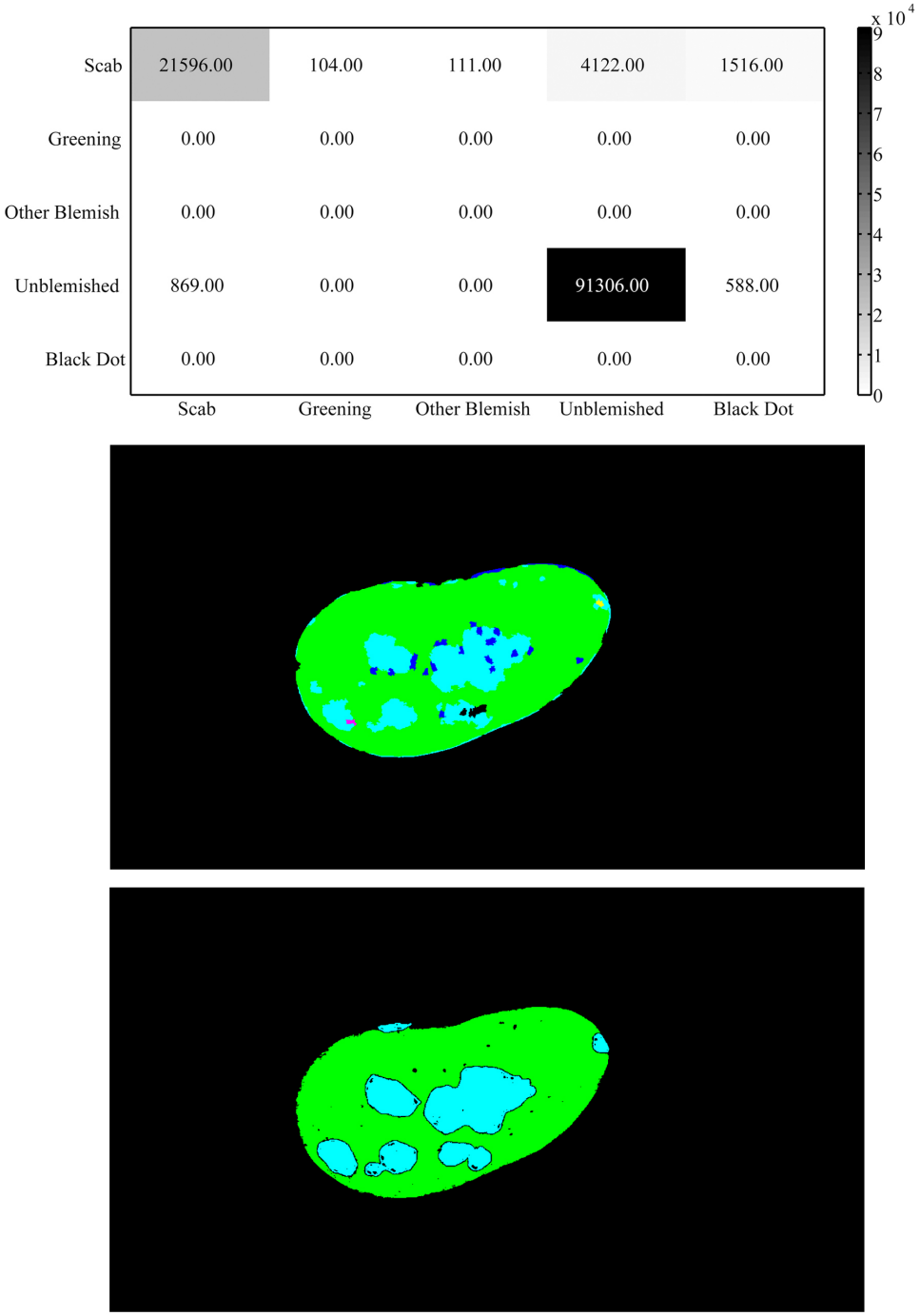


Figure 3: Top: Confusion matrix for the second input image; Middle: Classification results produced by the QA-TADD system; Bottom: Ground truth

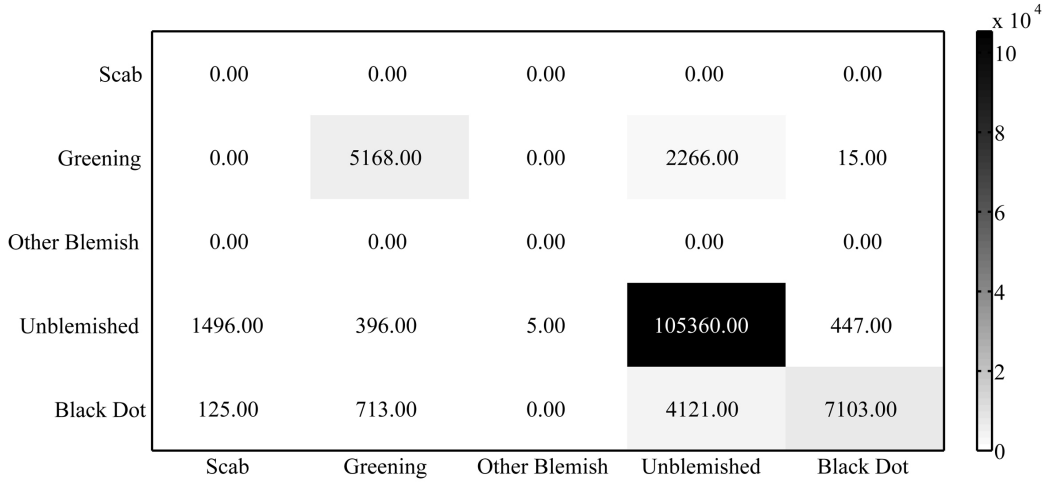


Figure 4: Confusion matrix for the seventh input image

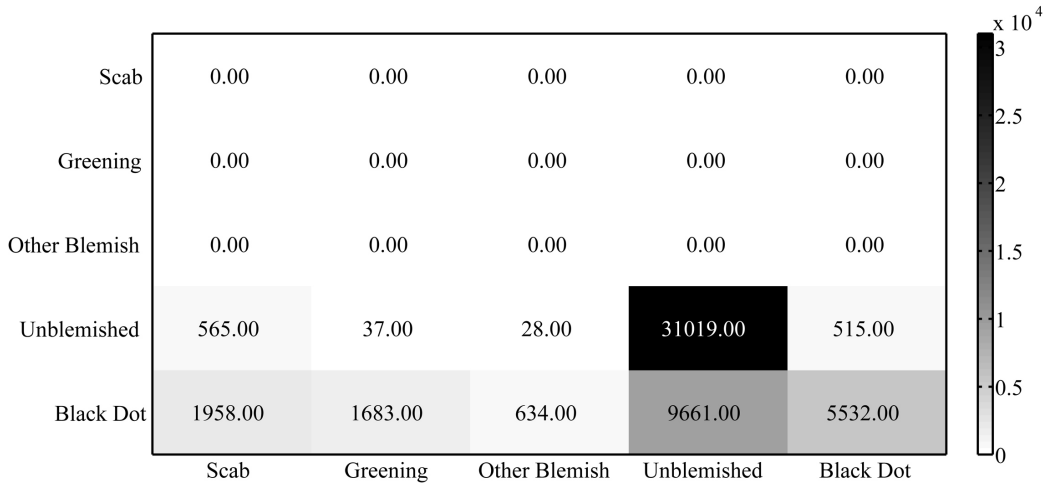


Figure 5: Confusion matrix for the eleventh input image

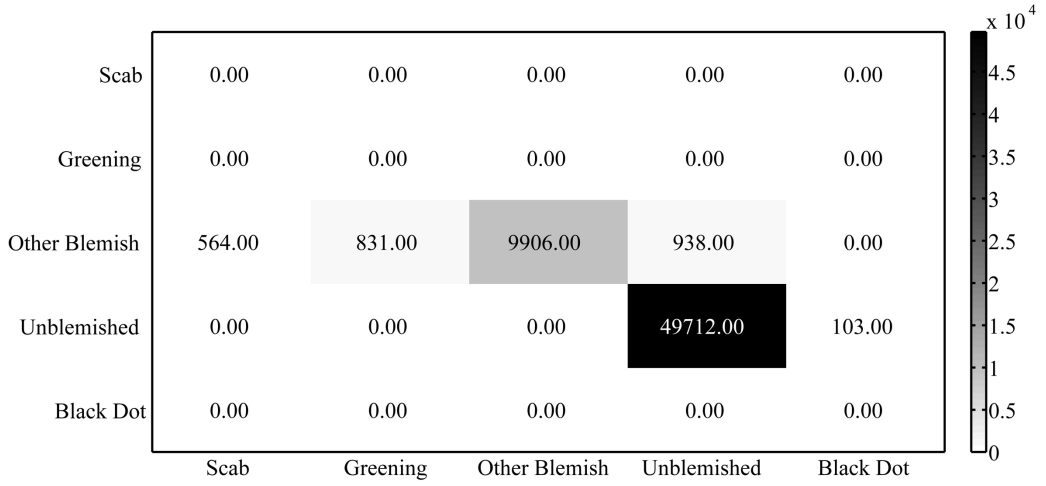


Figure 6: Confusion matrix for the twelfth input image

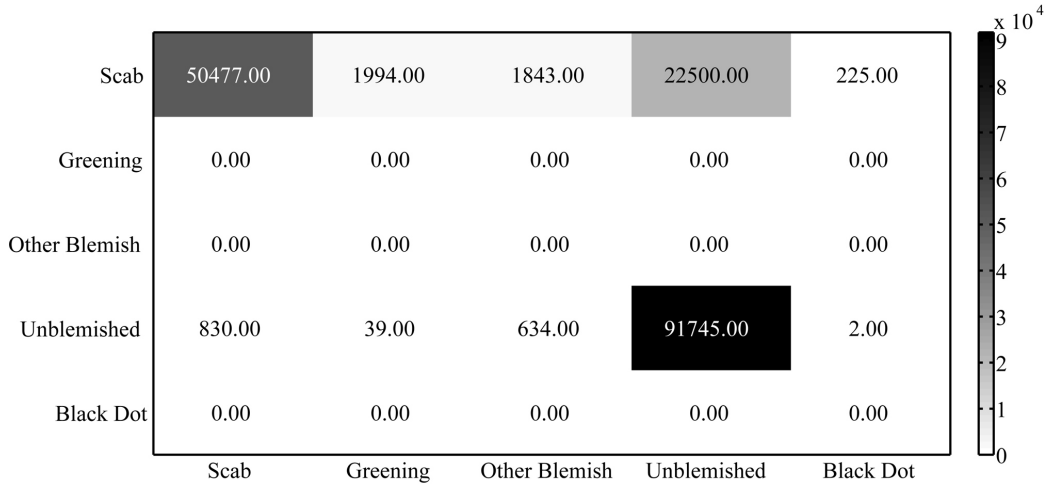


Figure 7: Confusion matrix for the twenty-first input image

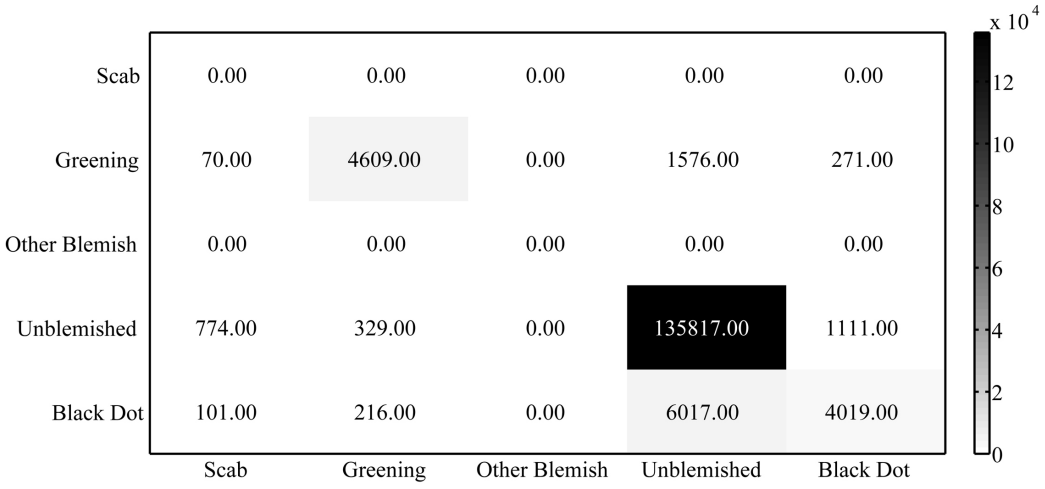


Figure 8: Confusion matrix for the thirty-fourth input image

TSB-funded Project 'TADD' - Trainable vision-based anomaly detection and diagnosis Technical Report for October 2014

Ran Song and Tom Duckett

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

Abstract

In October 2014, we developed a specific version of Label-TADD software to detect the position of a label where it is printed on the pack. Compared to the original Label-TADD software aiming at handling packs where labels are pasted onto the pack, the new version removes redundant functionalities to accelerate the implementation. We also newly introduced a 'mask' strategy to minimise the influence of reflection which, as is demonstrated with real data, is the most difficult issue and the major reason causing failures. On the other hand, we also evaluated the QA-TADD using the white potato dataset where we updated the codes for computing the confusion matrix by adding one more class: Silver Scurf, a common potato blemish.

1. Label-TADD for detecting printed label

Our original Label-TADD system aims at detecting label position where the label is pasted on the surface of the pack. In October, we bought some real food packs from TESCO and took photos of them using the camera on the Ishida machine. In these food packs, the labels are not pasted, but printed. That is to say, the label is a part of the pack. Therefore, we do not need to worry too much about whether the position of a label is correct or not since it is less likely to be wrong compared with the pasted labels. The only task is to detect its position. Such a straightforward observation means that we might be able to avoid inputting a 'pack with the label at the correct position' for training purpose. And, it is not necessary to calculate

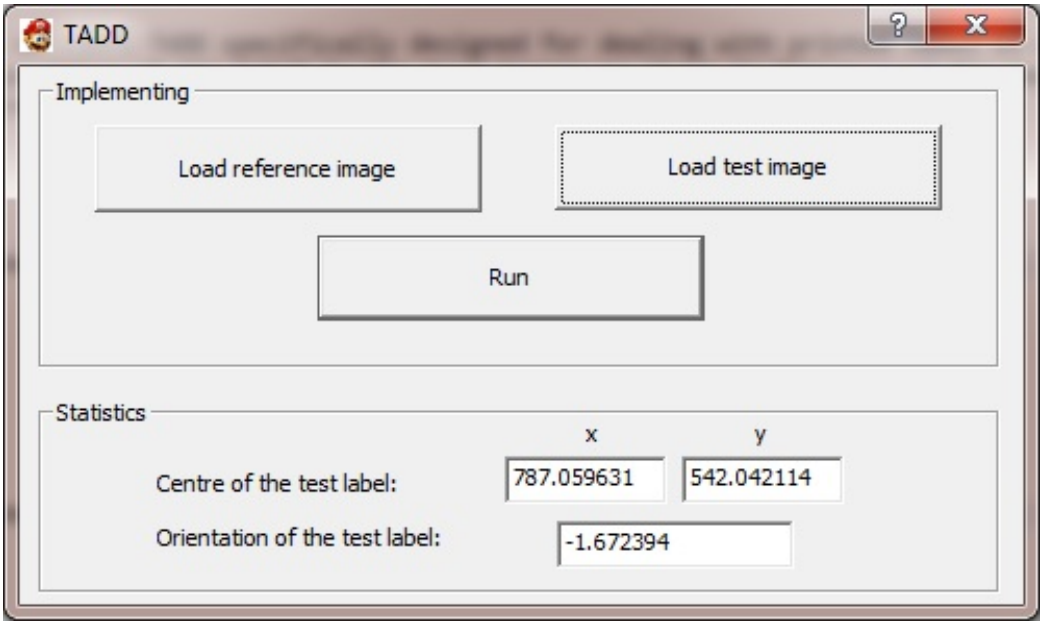


Figure 1: Interface of the new Label-TADD software

the position and the orientation errors and the implementation can thus be accelerated. We expect to confirm such an assumption with the Ishida people in the review meeting.

As shown in Figure 1, the version of Label-TADD specifically designed for dealing with printed label is quite simple. We just need to load a reference image without any printed label. Then we can load test images and the software will detect the position of the label in the test images as shown in Figure 2 and 3 where the reference images are produced by removing the label using Photoshop. Please note that in Figure 2 and 3, the textures of the ham and the chicken in the two packs are different. In practice, the reference image can be obtained by photographing a pack without any printed label.

To cope with strong reflection, such as the example shown in Figure 4, we introduce a 'mask' strategy. A mask image is a black and white image where the impossible positions of the label are filled with black. Since our Label-TADD is based on computing the difference image between the reference image and the registered test image, strong reflection could confuse the software where is the expected difference. Figure 5 shows some intermediate images with regard to the difference image. It can be seen that due to the

reflection, the different texture of the food inside the pack and the distortion of the pack, some undesired difference regions occur. In this case, without any prior knowledge, the software will randomly pick one region as the label region, which leads to the incorrect result shown in Figure 6. In contrast, Figure 7 shows a correct detection with the help of the mask image as shown in Figure 8.

According to our tests, if the reflection is not too much, it is not necessary to introduce the mask. Thus, it is vital to make sure that the image acquisition is of high quality and avoid introducing too much reflection while maintaining a sufficient brightness. If such an acquisition is hard to deliver, we should apply a 'strong' mask image where most of the image is filled with black. Note that since we register the test image towards the reference image, the mask image can be easily produced referring to the reference image.

If the final task of the software is to just read out the date and price information on the label, we do not need to compute and process the difference image. In this case, reflection will not be a problem. First, we manually draw a bounding box around the label region in the reference image (it can be a printed pack but need to be laid horizontally) and this only needs to be done once. Then, the software will automatically register any input test image into the coordinate system of the reference image. Finally, in the registered test image, we just need to cut out the region defined by the same bounding box and directly send it to OCR.

In summary, the tests using real food packs demonstrate that the core algorithm of the software, the registration, is quite reliable and is generally robust to reflection and the visual difference caused by the food inside the pack. In the tests we used 12 food packs to generate 6 reference images and 31 test images. There are 5 failed cases among 31 tests although 3 out of these 5 failed cases can be corrected by introducing a stronger mask image.

2. QA-TADD Evaluation

In October, we tested the QA-TADD system using 102 white potato images. I developed the MATLAB codes for computing the confusion matrix with respect to 6 different classes, namely Scab, Greening, Other blemish, Unblemished, Black dot and Sliver Scurf. Figure 9 shows the average confusion matrix where the training relied only on a tiny amount of superpixel-wise data. The average confusion matrix is based on the confusion matrices of all 102 test images. Basically it denotes that the black dots and the sliver scurf

are hard to be differentiated. Please refer to Hossein's report for the details of the evaluation results.

Figure 2: Implementation of the Label-TADD.

Figure 3: Implementation of the Label-TADD.

Figure 4: Reflection can cause significant visual differences between packs.

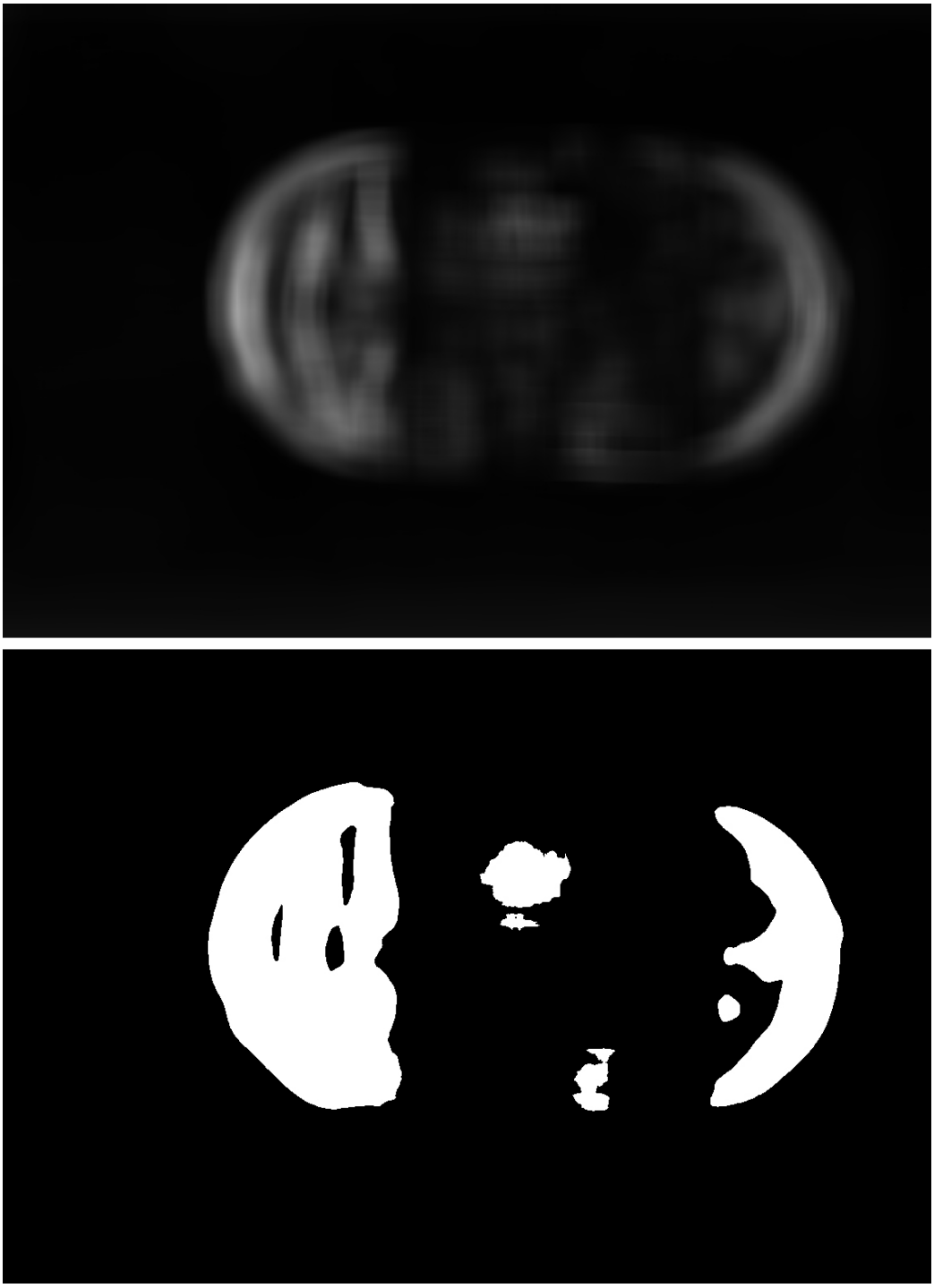


Figure 5: Reflection confuses the software.

Figure 6: Incorrect label detection

Figure 7: correct label detection

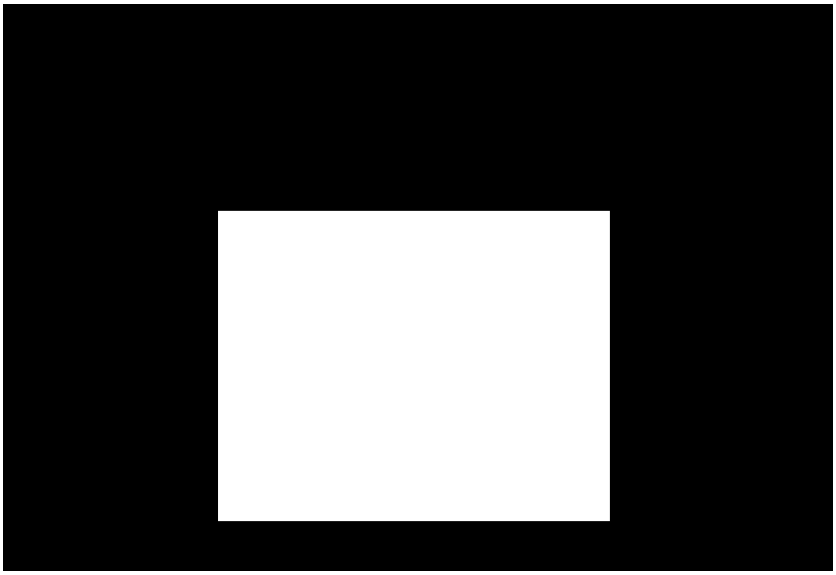


Figure 8: A mask image

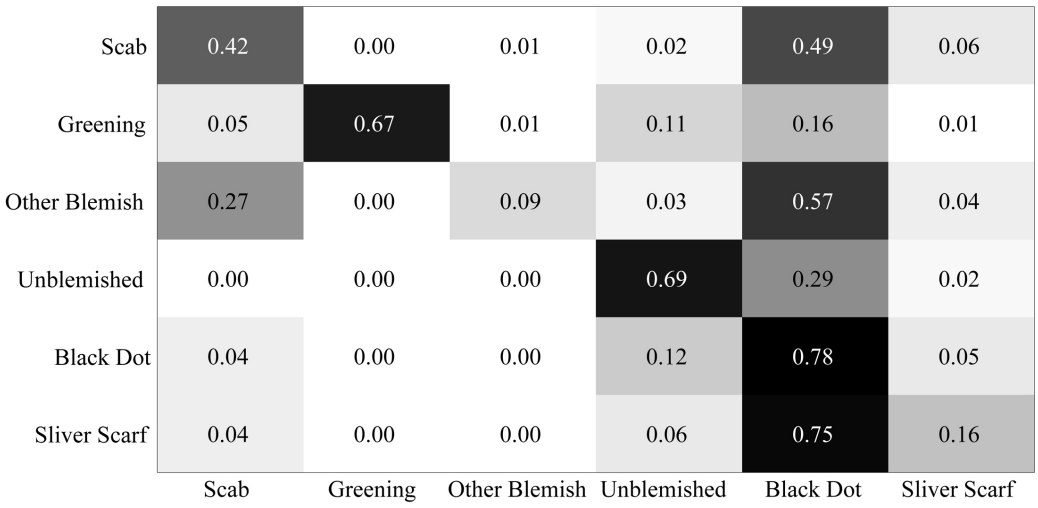


Figure 9: Confusion matrix for the classification of 102 white potato images

TSB-funded Project 'TADD'- Trainable vision-based anomaly detection and diagnosis Technical Report for August 2014

Hossein Malekmohamadi and Tom Duckett

Chapter 1

Introduction

This is the August 2014 report for the project TADD. The core topic of this report is software development for QA-TADD. I will be away on holidays from 15 to 31 August 2014, consequently this report covers research and development in QA-TADD for the first two weeks of August. The job description for this month is taken from the objectives of QA-TADD for Q7 which is implementing additional features to distinguish class types that are misclassified frequently.

Chapter 2

QA-TADD Development

In the last month (July 2014), we achieved the goal to enable QA-TADD work with high resolution images taken from DSLR camera. Since we get enough details from high resolution images to distinguish between similar blemish types it is time to focus on image feature extraction. The candidate features are Local Binary Patterns (LBP) [1] and Gabor filters [2, 3, 4].

We have implemented LBP and Gabor filter C++ codes. Examples are shown in Figures 2.1 to 2.3. Currently, this code extracs the features and saves the results in to the hard disk, but these codes will be integrated into TADD classifier when I come back in September. Also, we need to investigate parameters of each feature, e.g. number of neighbouriung pixels in LBP or wave length in Gabor as current code gives the freedom to end user to give some numeric values for each parameter. This can be done when we have the results of the confusion matrix of closely related blemish types.

Figure 2.1: Example of QA-TADD input.

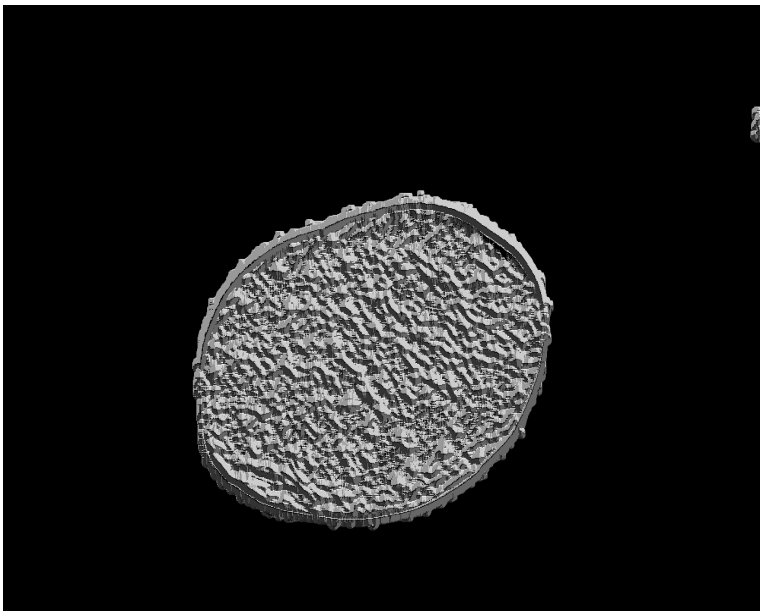


Figure 2.2: LBP map of the input to QA-TADD (shown in Figure 2.1).

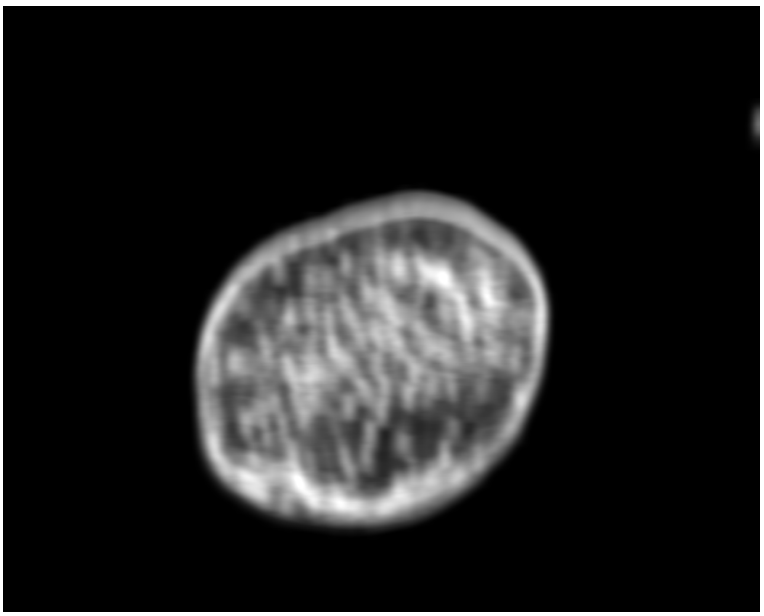


Figure 2.3: Gabor filtered image of the input to QA-TADD (shown in Figure 2.1).

Chapter 3

Conclusions

In this August 2014 report, we have written two feature extraction codes: LBP and Gabor filter. They have been added to image pre-processing part of QA-TADD. We may need first or higher order statistics of each of the extracted features before feeding them in to Adaboost algorithm.

Bibliography

[1] T. Ojala, M. Pietikinen, and D. Harwood (1994). Performance evaluation of texture measures with classification based on Kullback discrimination of distributions. Proceedings of the 12th IAPR International Conference on Pattern Recognition (ICPR 1994), vol. 1, pp. 582 - 585.

[2] Daugman, J. (1980). Two-dimensional analysis of cortical receptive field profiles. Vision Research, 20, 846856.

[3] Daugman, J. (1985). Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. Journal of the Optical Society of America-A, 2(7), 11601169.

[4] Gabor, D. (1946). Theory of communication. Journal of the Institute of Electrical Engineers, 93, 429457.

TSB-funded Project 'TADD'- Trainable vision-based anomaly detection and diagnosis Technical Report for September 2014

Hossein Malekmohamadi and Tom Duckett

Chapter 1

Introduction

This is the September 2014 report for the project TADD. The core topic of this report is software development for QA-TADD. The job description for this month is taken from the objectives of QA-TADD for Q7 which is implementing additional features to distinguish class types that are misclassified frequently. Moreover, we had Ensenso camera demo in Lincoln as you can see in the second part of this report.

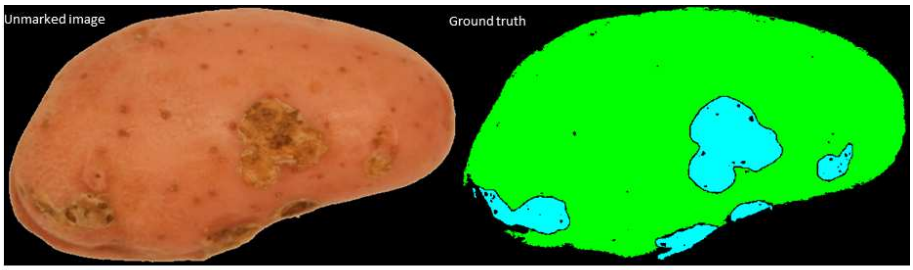
Part I

September Report

Chapter 2

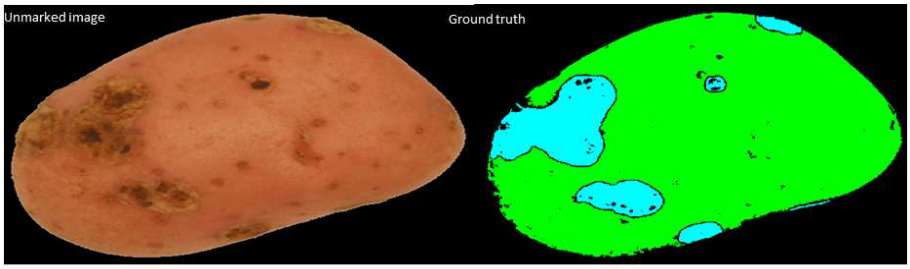
QA-TADD Development

In the last month (August 2014), we implemented LBP and Gabor filter C++ codes separately from QA-TADD. These codes were able to extract the features and save the results in to the hard disk. In this month, these features have been integrated into QA-TADD projects: cSLIC, TADDengine and QtTADD. First and higher orders statistics of filtered images (LBP or Gabor) have been used in QA-TADD. Afterwards, we have done some experiments to see the effects of these new features in distinguishing some blemish types when they were hard to classify. Figures 2.2 to 2.5 and 4.1 are some examples comparing the QA-TADD with basic features and basic features+LBP+Gabor, and total performance is shown in Table 2.1. We used 41 images of red potatoes to test QA-TADD which was trained with a saved classifier consisting of 8 images. Superpixel selection for this saved classifier was interactvie and it was reported in the last quarterly report.



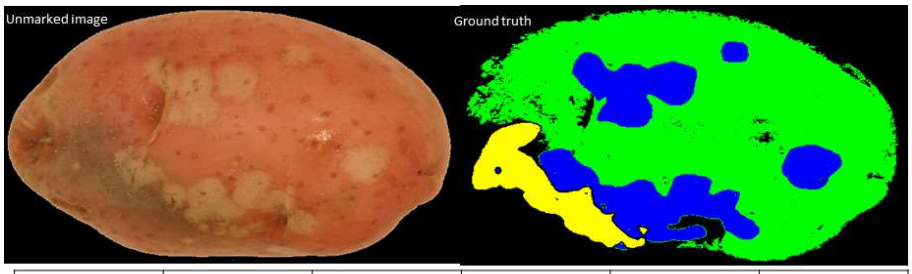
Actual/Detected	Scab	Greening	Other	Healthy	Black Dot
Scab	0.776, 0.75	0.009, 0.009	0.034, 0.04	0.155, 0.176	0.026, 0.025
Greening	0, 0	0, 0	0, 0	0, 0	0, 0
Other	0, 0	0, 0	0, 0	0, 0	0, 0
Healthy	0.015, 0.01	0.002, 0	0, 0	0.960, 0.974	0.023, 0.016
Black Dot	0, 0	0, 0	0, 0	0, 0	0, 0

Figure 2.1: Example 1 of QA-TADD result with basic features and with added features. Left image is the unmakred image and right image is the ground truth image. Left numbers are for the QA-TADD with basic features whilest the right numbers are from basic features plus two additional features.



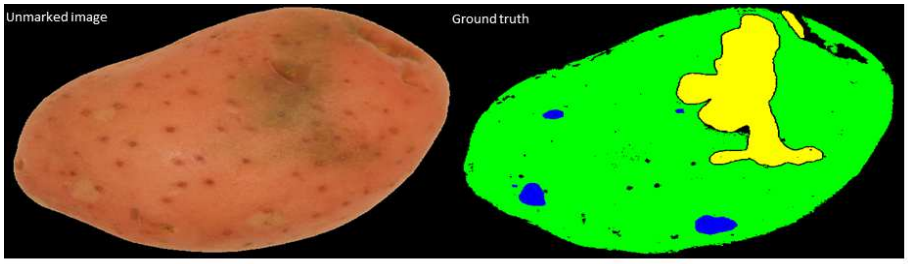
Actual/Detected	Scab	Greening	Other	Healthy	Black Dot
Scab	0.72, 0.738	0.008, 0.009	0.028, 0.022	0.244, 0.226	0, 0.005
Greening	0, 0	0, 0	0, 0	0, 0	0, 0
Other	0, 0	0, 0	0, 0	0, 0	0, 0
Healthy	0.02, 0.02	0, 0	0, 0	0.964, 0.972	0.016, 0.008
Black Dot	0, 0	0, 0	0, 0	0, 0	0, 0

Figure 2.2: Example 2 of QA-TADD result with basic features and with added features. Left image is the unmakred image and right image is the ground truth image. Left numbers are for the QA-TADD with basic features whilest the right numbers are from basic features plus two additional features.



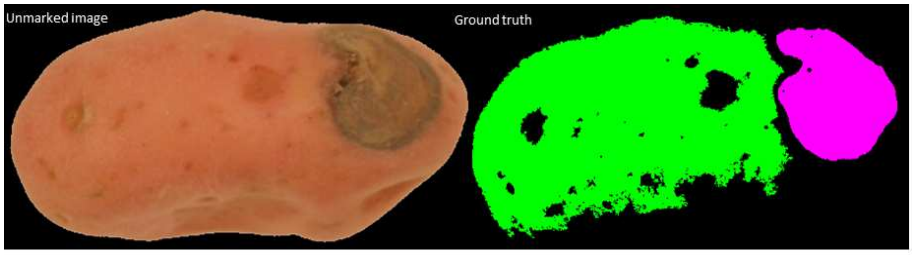
Actual\Detected	Scab	Greening	Other	Healthy	Black Dot
Scab	0, 0	0, 0	0, 0	0, 0	0, 0
Greening	0.023, 0.039	0.210, 0.372	0.646, 0.571	0.121, 0.018	0, 0
Other	0, 0	0, 0	0, 0	0, 0	0, 0
Healthy	0.006, 0.006	0.006, 0.003	0.002, 0.002	0.98, 0.982	0.006, 0.007
Black Dot	0.045, 0.082	0.056, 0.077	0.102, 0.085	0.372, 0.309	0.425, 0.447

Figure 2.3: Example 3 of QA-TADD result with basic features and with added features. Left image is the unmakred image and right image is the ground truth image. Left numbers are for the QA-TADD with basic features whilest the right numbers are from basic features plus two additional features.



Actual/Detected	Scab	Greening	Other	Healthy	Black Dot
Scab	0, 0	0, 0	0, 0	0, 0	0, 0
Greening	0.053, 0.004	0.242, 0.027	0.088, 0.070	0.533, 0.600	0.084, 0.299
Other	0, 0	0, 0	0, 0	0, 0	0, 0
Healthy	0.016, 0.001	0.001, 0.011	0, 0	0.964, 0.976	0.019, 0.012
Black Dot	0, 0.02	0, 0	0, 0	0.51, 0.707	0.49, 0.273

Figure 2.4: Example 4 of QA-TADD result with basic features and with added features. Left image is the unmakred image and right image is the ground truth image. Left numbers are for the QA-TADD with basic features whilest the right numbers are from basic features plus two additional features.



Actual\Detected	Scab	Greening	Other	Healthy	Black Dot
Scab	0, 0	0, 0	0, 0	0, 0	0, 0
Greening	0, 0	0, 0	0, 0	0, 0	0, 0
Other	0.047, 0.042	0.068, 0.101	0.81, 0.82	0.075, 0.037	0, 0
Healthy	0, 0	0, 0.002	0, 0	0.998, 0.997	0.002, 0.001
Black Dot	0, 0	0, 0	0, 0	0, 0	0, 0

Figure 2.5: Example 5 of QA-TADD result with basic features and with added features. Left image is the unmakred image and right image is the ground truth image. Left numbers are for the QA-TADD with basic features whilest the right numbers are from basic features plus two additional features.

Table 2.1: QA-TADD result with basic features and with added features. Left numbers are for the QA-TADD with basic features whilest the right numbers are from basic features plus two additional features.

Actual \ Detected	Scab	Greening	Other	Healthy	Black Dot
Scab	0.682, 0.703	0.021, 0.024	0.023, 0.013	0.246, 0.226	0.028, 0.034
Greening	0.074, 0.044	0.300, 0.288	0.085, 0.108	0.512, 0.547	0.029, 0.013
Other	0.039, 0.055	0.218, 0.095	0.529, 0.592	0.213, 0.256	0.001, 0.002
Healthy	0.008, 0.008	0.004, 0.004	0.001, 0	0.979, 0.982	0.008, 0.006
Black Dot	0.027, 0.022	0.041, 0.041	0.025, 0.018	0.540, 0.573	0.367, 0.346

Chapter 3

Conclusions

In this September 2014 report, we have integrated two features (LBP and Gabor filter) into QA-TADD and we initialised their parameters based on several experiments. They have been added to Gslic, QtTADD and TAD-Dengine parts of QA-TADD. Afterwards, we compared results of blemish type classification with these additional features with QA-TADD using just the basic features. Adding these two features led to better performance detecting healthy, uknown blemish and scab in red potatoes. Further analysis on white potatoes is carried forward to the next month.

Part II

Ensenso Demo

Chapter 4

Introduction

On 04/09/2014, we had the chance to see Ensenso N20 3D camera in action at UoL. Nathaniel Hofmann form Multipix came to UoL to demo the camera. The Ensenso stereo 3D cameras work according to the projected texture stereo vision principle [1]. Each model has two integrated CMOS sensors and a projector that casts a random point pattern onto the object to be captured. The key advantage of the pattern is that it also works in multi-camera mode and can capture images of surfaces that have virtually no texture at all. The compact and robust aluminum housing of the cameras with lockable GPIO connector for trigger and flash underline the suitability of the cameras for industrial use [2]. The cameras are pre-calibrated and come with an MVTec HALCON interface as well as an object-oriented API (C++, C# / .NET). N20 has numerous advantages[2]:

- Versatile and flexible GigE interface
- Compact, robust aluminum housing
- Integrated global shutter CMOS sensors and pattern projector with blue LEDs
- Up to 30 frames per second at full resolution and 64 disparity levels
- Designed for working distances of up to 3,000 mm (N20) and variable picture fields. This model has a working distance of 380-1900 mm
- Output of a single 3D point cloud with data from all cameras used in multi-camera mode
- Live composition of the 3D point clouds from multiple viewing directions

- Projected texture stereo vision process for capturing untextured surfaces
- Capture of both stationary and moving objects
- Free software package with driver and API for Windows and Linux
- One software package supports USB and GigE models
- HALCON, C, C++ and C# sample programs with source code
- Pre-calibrated and therefore easy to set up
- Integrated function for robot hand-eye calibration with calibration plate
- Integration of uEye industrial cameras on the software side, for example, to capture additional color information or barcodes
- Subsampling and binning for flexible data and frame rates

An image with ready meal and potato samples and camera setup is shown in Figure 4.1. Working distance in our demo is set to 490 mm.

Figure 4.1: Ensenso N20 setup.

Chapter 5

3D Data Collection

In this section, we present 5 different experiments with ready meals and raw food.

5.1 Experiment 1: Indian Snack

In Figures 5.1 to 5.3, an Indian snack is shown with all packaging, without packaging and its empty tray. There are some regions that 3D information cannot be retrieved due to the amount of reflection of N20 blue flashlight. However, this can be fixed with increasing working distance or decreasing light power.

5.2 Experiment 2: Indian Meal

In Figures 5.4 to 5.6, an Indian meal is shown with all packaging, without packaging and its empty tray. Likewise, there are some regions that 3D information in them cannot be retrieved. In Figure 5.6, an empty tray with different sections is retrieved despite having many reflected areas.

5.3 Experiment 3: Italian Meal

In Figures 5.7 to 5.9, an Italian meal is shown with all packaging, without packaging and its empty tray. Please note that 3D information of some regions cannot be retrieved due to the reflection of the blue flashlight.

Figure 5.1: Indian snack with all packaging.

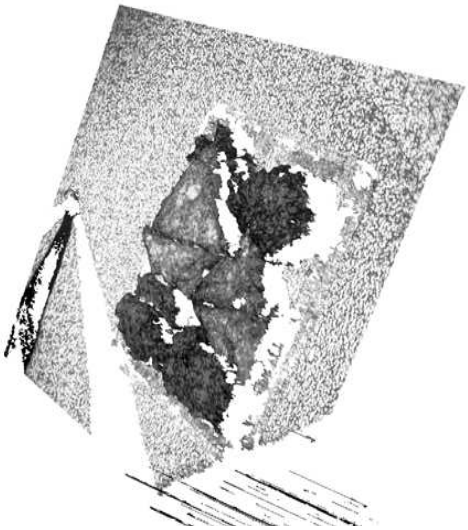


Figure 5.2: Indian snack without packaging.

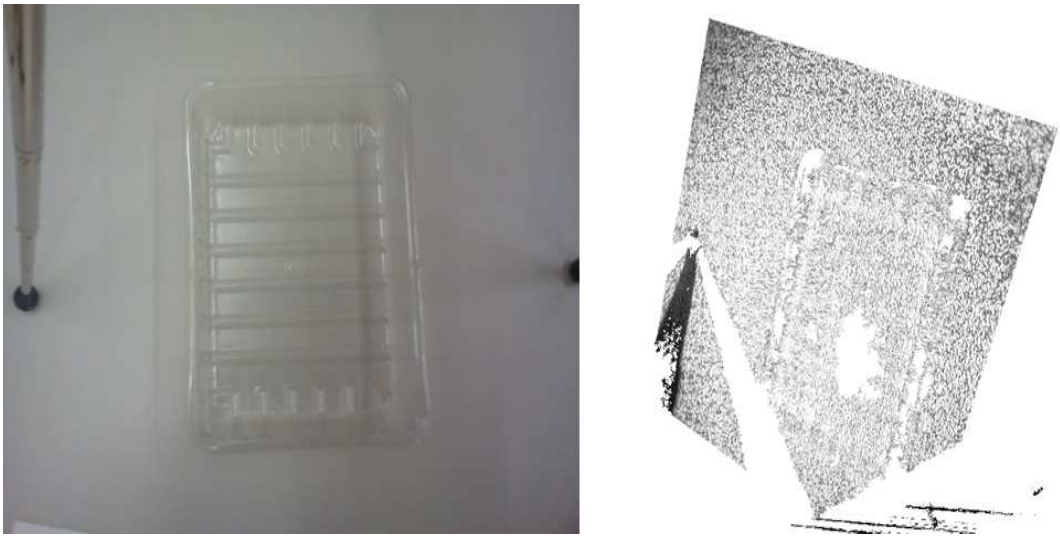


Figure 5.3: Empty tray of the Indian snack.

Figure 5.4: Indian meal with all packaging.

Figure 5.5: Indian meal without packaging.

Figure 5.6: Empty tray of the Indian meal.

Figure 5.7: Italian meal with all packaging.

Figure 5.8: Italian meal without packaging.

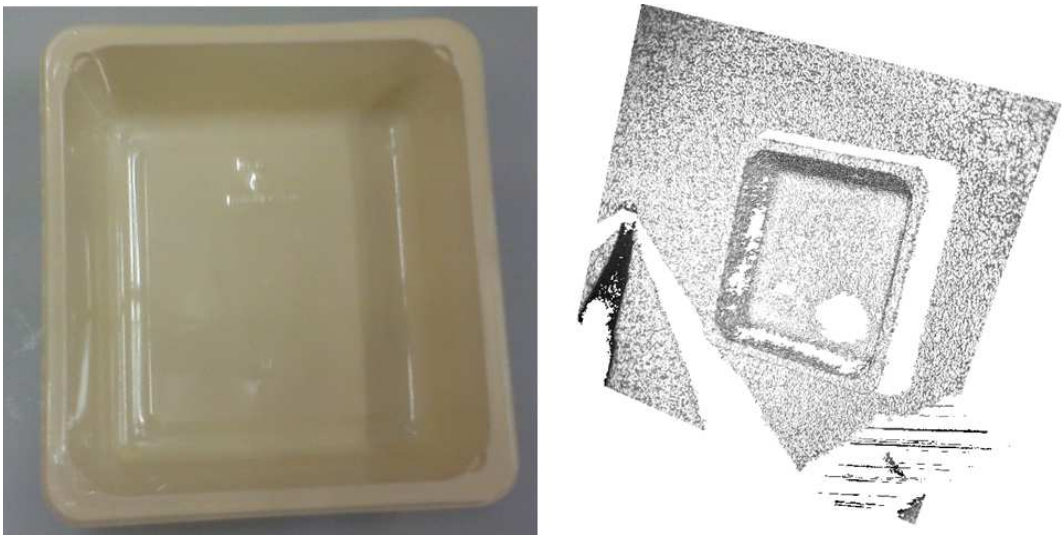


Figure 5.9: Empty tray of the Italian meal.

5.4 Experiment 4: Spanish Meal

In Figures 5.10 to 5.12, a Spanish meal is shown with all packaging, without packaging and its empty tray. Likewise, there are some regions that 3D information in them cannot be retrieved due to the amount of reflection of N20. Food components can be separated based on their volume and shape as shown in Figure 5.11.

Figure 5.10: Spanish meal with all packaging.

Figure 5.11: Spanish meal without packaging.

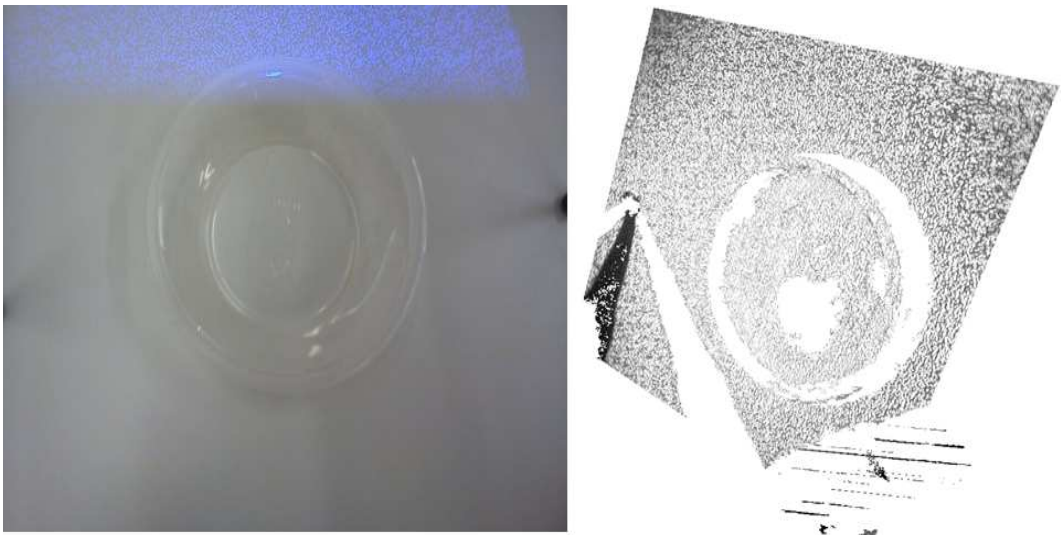


Figure 5.12: Empty tray of the Spanish meal.

5.5 Experiment 5

In Figures 5.13 and 5.14, a torn bag of potato and one potatoes are shown respectively. In 3D model of Figure 5.14, background is removed and you can see the defect on the potato skin where we can achieve 0.5mm accuracy.

Figure 5.13: A bag of potatoes where different heights are recognisable as well as some skin defects.

Figure 5.14: A potato with skin defects and its 3D model.

Chapter 6

Conclusions

In this report, we presented some experiments with the Ensenso N20 3D camera. Benefits of this camera are operational working distance for the current TADD setup, accuracy up to 0.5mm, GigE connection and SDK in MVTec HALCON interface as well as an object-oriented API (C++, C# / .NET). Moreover, the results show that it is possible to acquire high quality 3D images of food products with the Ensenso N20.

Bibliography

[1] Konolige, Kurt. "Projected texture stereo." Robotics and Automation (ICRA), 2010 IEEE International Conference on. IEEE, 2010.

[2] http://en.ids-imaging.com/store/produkte/kameras/ensenso-n20-3d-gige/show/all.html

TSB-funded Project 'TADD'- Trainable vision-based anomaly detection and diagnosis Technical Report for October 2014

Hossein Malekmohamadi and Tom Duckett

Chapter 1

Introduction

This is the October 2014 report for the project TADD. The core topic of this report is extensive analysis of multiple-blemish classifier with the additional features.

Chapter 2

QA-TADD Development

In the last month (September 2014), we analysed QA-TADD to classify different belmish types in red potatoes with the added features: LBP and Gabor filters [1, 2, 3, 4]. We have integrated LBP and Gabor filter C++ codes into QA-TADD. In the first set of experiments on the white potato database (102 images) we just use the basic features of TADD to distinguish between 6 classes. In the second experiment, we study the effect of neighbouring information in classification results by increasing the number of neighbourhood pixels in LBP from 4 to 16. Finally, we try to achieve an optimum settings for the QA-TADD with combination of bigger superpixel size and higher number of neighbouring pixels. In these experiments, we used interactvie training scheme.

We divided 102 images in to 3 bins (statistical analysis has been done for 6 classes): (1) Basic bin that has 10 images with at least 2 blemish types each covers more than 40% of potato area; (2) Second bin has 10 images each has at least 3 belmish types for more than 20% of the potato area and (3) the rest 82 images are put in the last bin. Users are shown the ground truth images of the basic bin and they can select superpixels they want to train. All images of basic bin are used. Afterwards, users can pick at least 2 images from the second bin and the rest of the images from the last bin. Users are advised to pick maximum 20 images to train QA-TADD to have a fast interactive traning and testing session. Ground truth images are not shown in this part. Based on this permutation, we created an interactive training data consisting of 13 images. Consequently, the results below are for an interactive traning procedure while the saved classifier has 13 images. Please note that, classifier is not tested on its own traning data while selecting superpixels to avoid deselction/reselection process. The number of superpixels selected in training data and the number of pixels tested in test data are shown in

Tables 2.1 and 2.2 resepctively. Please note that in all tables and figures in this report we use these acronyms:

- BD= Black Dot
- SC= Scab
- HE= Healthy
- UN= Unknown blemish
- SS=Silver Scurf
- GR=Greening
- BT= Blemish Type
- IN= Image Number

Table 2.1: QA-TADD training data.

BT \ IN	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11	I12	I13	Sum
BD	40	29	0	24	0	0	89	43	75	0	46	68	151	565
SC	0	0	72	165	52	0	0	0	41	84	17	0	0	431
HE	18	96	43	87	124	88	145	93	72	136	37	56	101	1096
UN	0	0	0	9	0	43	72	61	0	0	0	0	0	185
SS	49	50	82	24	63	0	0	29	0	89	14	46	0	446
GR	0	0	0	0	0	47	0	0	0	0	0	0	0	47

Table 2.2: Number of pixels in QA-TADD test data for each blemish type. We consider average superpixel size contains 100 pixels. And we use results from Table 2.1 here.

BT	Testing pixels	Estimated number of training pixels	Training to testing ratio
BD	2447893	56500	0.023
SC	354203	43100	0.121
HE	4073505	109600	0.027
UN	116373	18500	0.159
SS	969588	44600	0.046
GR	274399	4700	0.017

2.1 Experiments with basic features

In this experiment, we use QA-TADD basic features as shown in Table 2.3. An example image with its ground truth are shown in Figures 2.1 and 2.2.

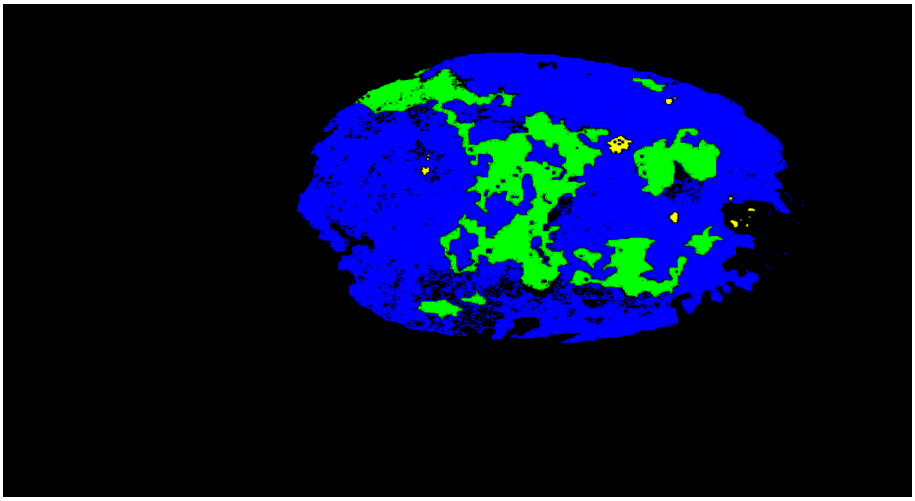


Figure 2.1: Ground truth image.

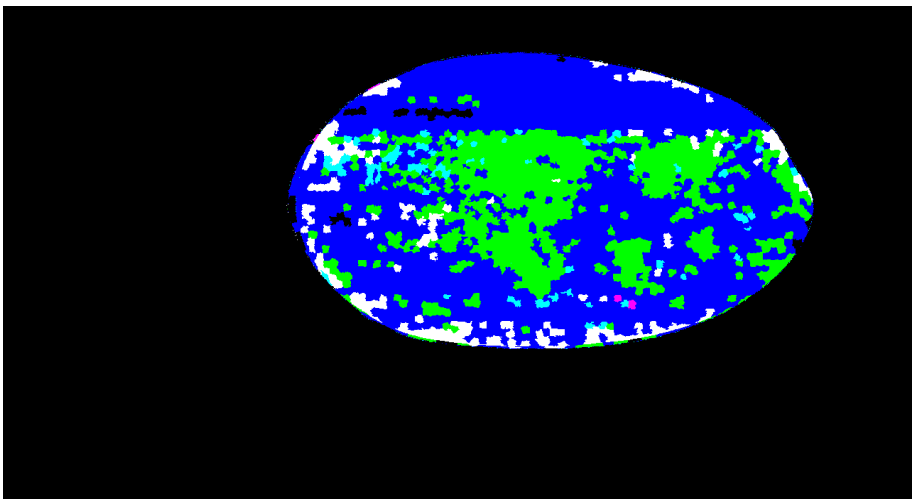


Figure 2.2: Result with basic features.

Table 2.3: QA-TADD results with basic features for 102 images.

Actual \ Detected	BD	SC	HE	UN	SS	GR
BD	0.783	0.045	0.122	0.001	0.052	0
SC	0.488	0.423	0.022	0.007	0.06	0
HE	0.287	0.002	0.687	0	0.017	0.007
UN	0.568	0.269	0.034	0.087	0.042	0
SS	0.745	0.036	0.059	0.001	0.159	0
GR	0.162	0.051	0.108	0.005	0.006	0.668

2.2 Experiments with added features

In this experiment, we use QA-TADD basic features and additional features as shown in Tables 2.4 and 2.5. In the first experiment we used LBP-4 and in the second onbe we used LBP-16. The main reason to increase neighbouring information is to distinguish more between black dot and silver scurf. Black dot is a kind of blemish that can occupy everal pixels much less than a superpixel size, hence it is neccessary to acquire more information for its neighbouring pixels as much as possible. Examples are shown in Figures 2.3 and 2.4.

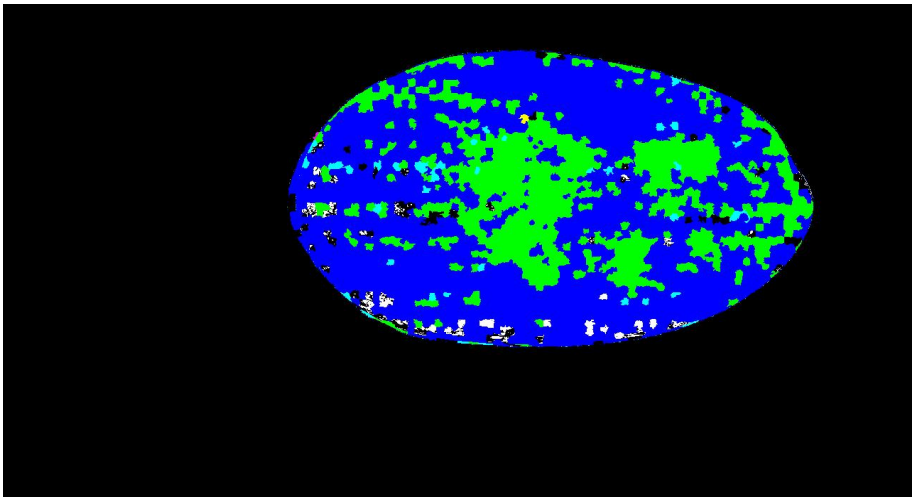


Figure 2.3: Result with additional features. LBP-4 is used here.

Table 2.4: QA-TADD results with basic features and additional features for 102 images. In this case we use 4-neghbouring pixels for LBP.

Actual \ Detected	BD	SC	HE	UN	SS	GR
BD	**0.807**	0.018	0.148	0.001	0.026	0
SC	0.754	0.131	0.019	0.001	0.095	0
HE	0.248	0	**0.748**	0	0	0.004
UN	0.814	0.034	0.028	0.039	0.085	0
SS	0.788	0.016	0.088	0	0.108	0
GR	0.209	0.006	0.116	0	0.004	0.665

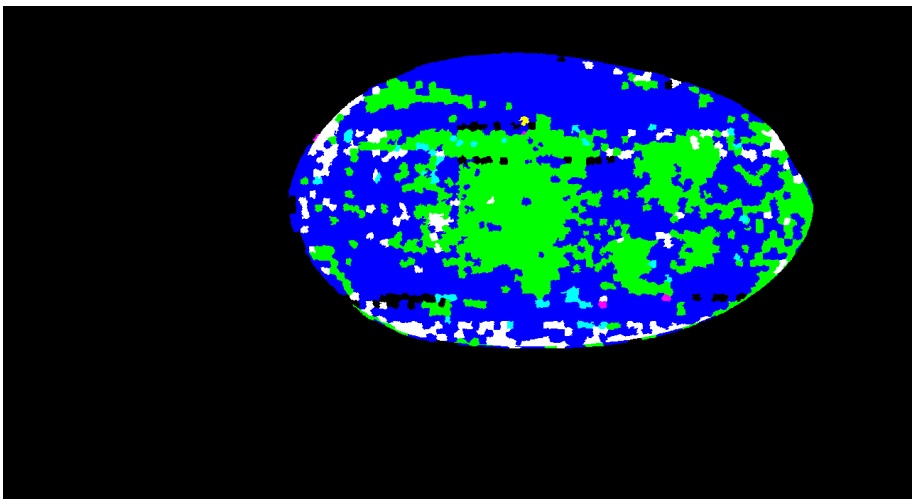


Figure 2.4: Result with additional features. LBP-16 is used here.

2.3 Optimum settings

In this experiment, we use QA-TADD basic features and additional features as shown in Table 2.6. We kept LBP-16 but increased superpixel wight factor from 0.1 to 0.125. This has led to having bigger superpixels. This helped the system to classify blemish types more accurately than any other settings we had earlier. One example is shown in Figure 2.5.

As shown in the experiments, increasing the number of neighbouring pixels in LBP with bigger superpixel size has led to better accuracy. There are some points to mention here. First, unknown blemish type is the hardest to distinguish as we have no information about them in the ground truth images and even if we use samples of them to trani QA-TADD we inserted noisy data

Table 2.5: QA-TADD results with basic features and additional features for 102 images. In this case we use 16-neghbouring pixels for LBP.

Actual \ Detected	BD	SC	HE	UN	SS	GR
BD	0.757	0.024	0.166	0.002	0.051	0
SC	0.568	0.394	0.015	0.002	0.021	0
HE	0.219	0.001	**0.766**	0	0.011	0.003
UN	0.807	0.087	0.021	0.043	0.042	0
SS	0.669	0.01	0.114	0	**0.207**	0
GR	0.173	0.034	0.106	0.011	0.011	0.665

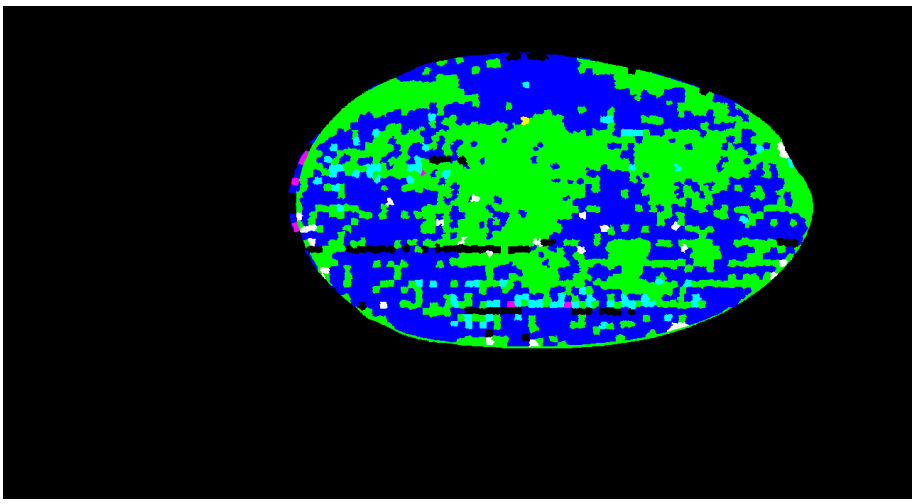


Figure 2.5: Result with additional features. LBP-16 and bigger superpixels are used here.

while training. Second, by increasing superpixel size and using LBP-16 we can increase accuracy in detecting black dot, scab and silver scurf. However, for an ideal system we may need to look at each image globally and detect if an image has higher chance of having black dot, scab or silver scurf. Then we will be able to change features weight in the learning algorithm accordingly. This feedback system will require another mahcine learning algorithm, e.g. SVM, to examine each image with a learnt data. Last, in all combinations QA-TADD has an acceptable accruacy in detecting greening and healthy potatoes. However, the optimum settings improvement is more noticable.

Table 2.6: QA-TADD results with basic features and additional features for 102 images. In this case we use 16-neghbouring pixels for LBP and bigger superpixel size. We changed superpixel weight from 0.1 to 0.125.

Actual \ Detected	BD	SC	HE	UN	SS	GR
BD	0.72	0.044	0.221	0	0.015	0
SC	0.385	**0.539**	0.054	0.002	0.019	0.001
HE	0.198	0.002	**0.764**	0	0.032	0.004
UN	0.645	0.135	0.117	**0.092**	0.011	0
SS	0.360	0.027	0.135	0.002	**0.476**	0
GR	0.106	0.02	0.099	0	0.004	**0.771**

Chapter 3

Conclusions

In this October 2014 report, we analysed QA-TADD capabilities in detecting different blemish types in white potatoes. It was shwon that by increasing the number of neighbouring pixels while extracting statistical features like LBP we can accommodate more information on certain blemish types. QA-TADD can even have better accuracy by small increase in superpixel size.

Bibliography

[1] T. Ojala, M. Pietikinen, and D. Harwood (1994). Performance evaluation of texture measures with classification based on Kullback discrimination of distributions. Proceedings of the 12th IAPR International Conference on Pattern Recognition (ICPR 1994), vol. 1, pp. 582 - 585.

[2] Daugman, J. (1980). Two-dimensional analysis of cortical receptive field profiles. Vision Research, 20, 846856.

[3] Daugman, J. (1985). Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. Journal of the Optical Society of America-A, 2(7), 11601169.

[4] Gabor, D. (1946). Theory of communication. Journal of the Institute of Electrical Engineers, 93, 429457.