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

August 2014

Abstract

This technical quarterly report is a summary of the work we have done in May, June and July 2014 for the project Trainable Vision-based Anomaly Detection and Diagnosis (TADD). We just integrate the corresponding 6 monthly technical reports produced by Dr Ran Song and Dr Hossein Malekmohamadi respectively.

TSB-funded Project 'TADD' - Trainable vision-based anomaly detection and diagnosis Technical Report for May 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 May 2014, I developed a user-friendly GUI (short for graphical user interface) for the label-TADD system. It integrates all functionalities of the previous command line TADD software and is specifically designed for performing large scale tests using a number of images saved on the disk. The entire layout of the software is consistent with the standard of modern industrial software applications. To further facilitate the users, some hints, reminders and messages are also added within the software.

1. About GUI

In computing, graphical user interface (GUI) is a type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, as opposed to text-based interfaces, typed command labels or text navigation. GUIs were introduced in reaction to the perceived steep learning curve of command-line interfaces (CLI), which require commands to be typed on the keyboard.

The actions in GUI are usually performed through direct manipulation of the graphical elements. Besides in computers, GUIs can be found in handheld devices such as MP3 players, portable media players, gaming devices, household appliances, office, and industry equipment. In May 2014, I developed a GUI for the label-TADD software. It is designed for testing the functionalities of the original command line-based label-TADD application in an easier way.

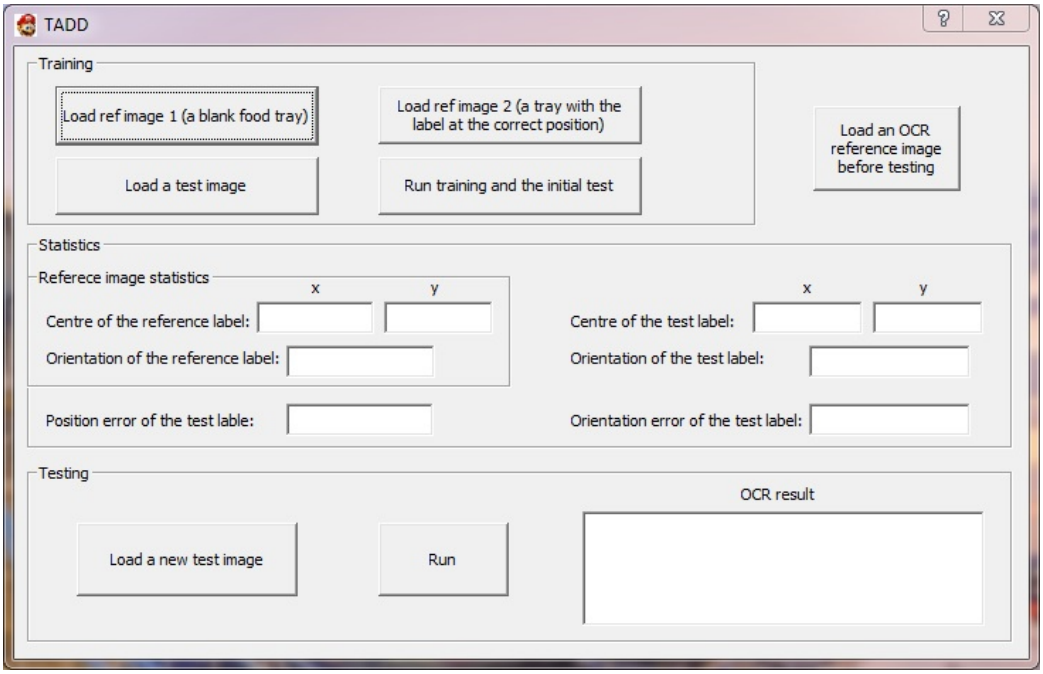

Figure 1: The GUI for label-TADD

2. Label-TADD GUI

The GUI is developed using MFC (short for Microsoft Foundation Class Library) which is a library that wraps portions of the Windows API in C++ classes. It is currently one of the mainstream tools for developing GUIs of commercial application software.

As shown in Figure. 1 the interface is composed of three major regions. The top region is the training region. The four buttons grouped in this region offer the functionalities for training the system and performing an initial test. The first button requires the user to load an image of a blank food tray as the first reference image. When clicking it, it will generate a standard Windows dialog box as shown in Figure. 2. And once a blank tray image has been selected, the system will show it in a new window. The window is adjustable and the software also enables the zooming-in/out functionalities when displaying images. This can be done through the buttons listed in the window (as shown in Figure. 3) or by scrolling the mouse wheel. Some other functionalities, such as browsing the image, saving the image and reading the XYRGB values of pixels are also available.

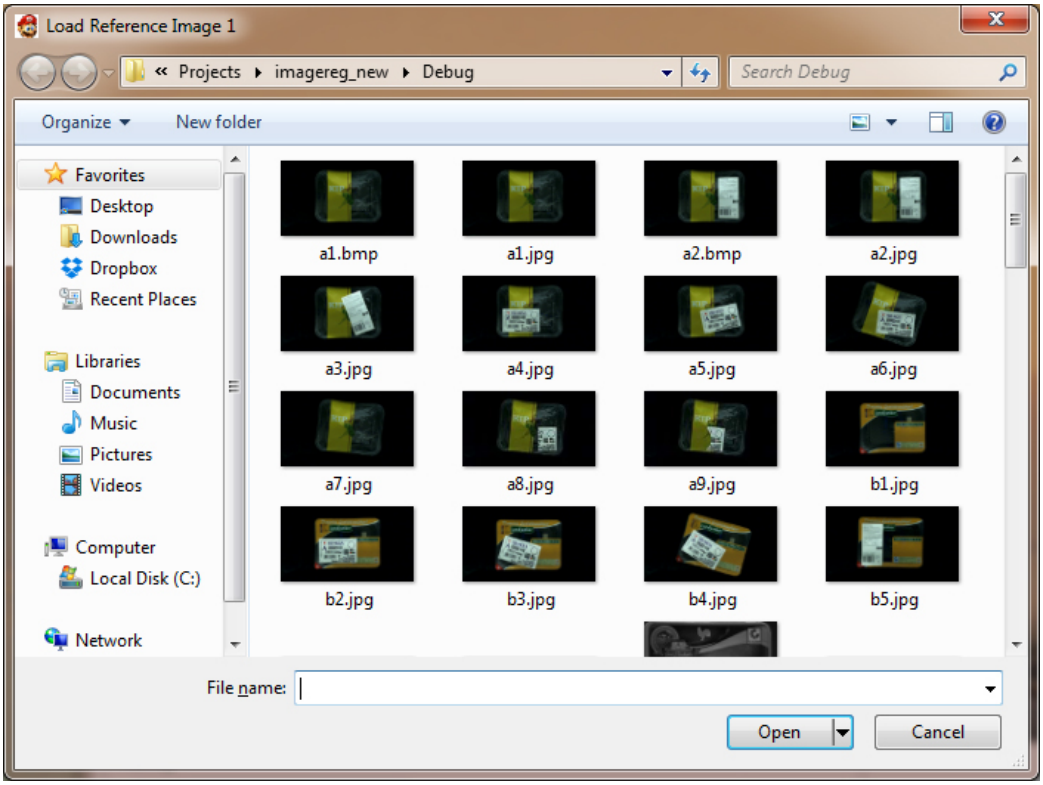

Figure 2: Load the first reference image for training

Once loading and showing the first reference image, the user can close the image window by pressing the 'Esc' key. Then the user needs to load the second reference image which is a food tray with the label at the correction position and a test image. Next, by clicking the 'Run training and the initial test' button, a new window titled 'Test Result' will pop out, as shown in Figure. 4. Certainly the user can save the resultant image by clicking the button in the toolbar. The default file name is Test Result_screenshot_dd.mm.yyyy.png where dd.mm.yyyy denotes the current date. By pressing any key on the keyboard, the window showing the result of the initial test will be closed and a new dialog box will pop up. It delivers a message: "If the initialisation result is not good, redo the training and initial test." By clicking the 'OK' button in the dialog box, the message will disappear and the user can decide what to do next: retraining or testing new images (where training is not needed any more). Meanwhile the GUI has been fed with some statistics in

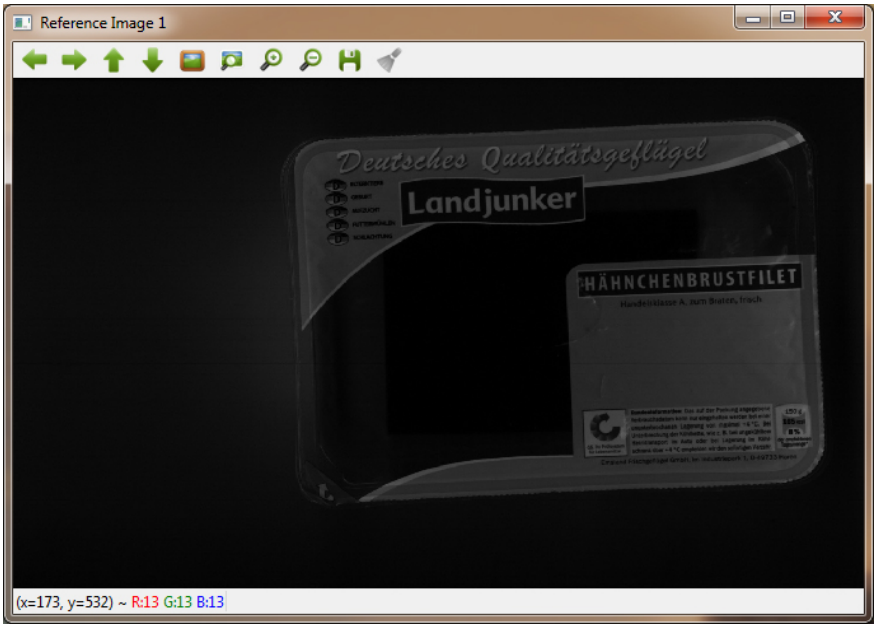

Figure 3: Show the reference image

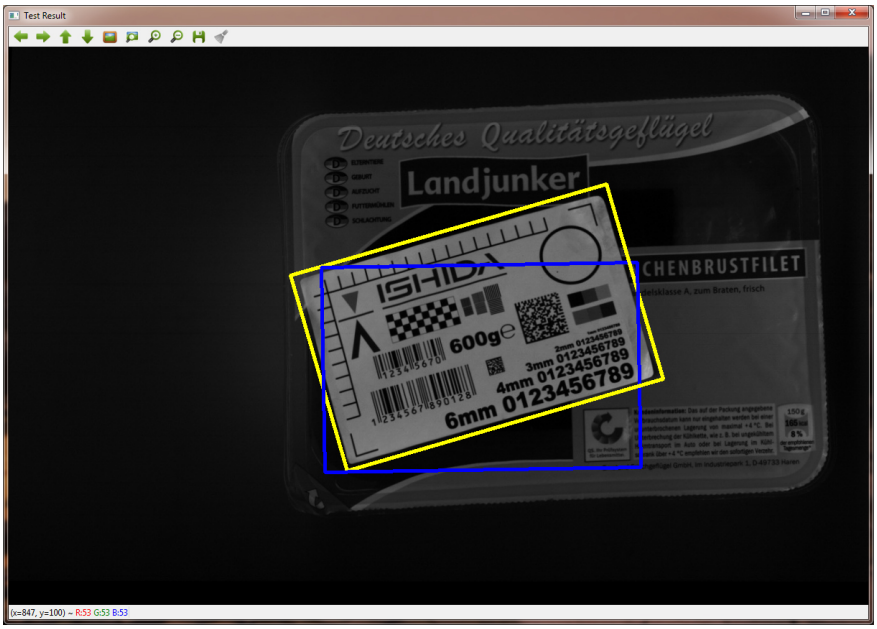

Figure 4: Show the result of initial test

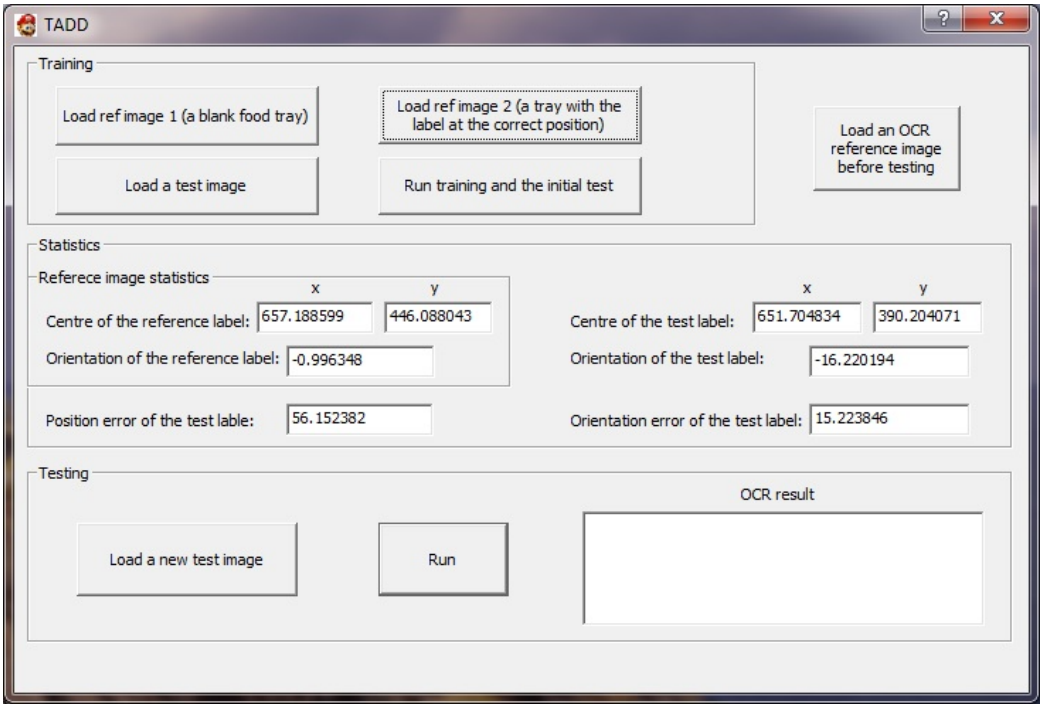

Figure 5: After the initial test, the 'Statistics' panel in the middle of the GUI is updated with some statistics.

the 'Statistics' panel, as shown in Figure. 5. There is a subpanel titled 'Reference image statistics'. It includes the XY coordinates of the centre of the label in the reference image and the orientation of the label. These statistics represent the correct position and orientation of the label (where it should be) and thus will remain unchanged in the following testing stage. The rest of the panel displays the information related to the label in the test image. Besides its position and orientation, the key information is the position and the orientation errors of the label. These statistics will be updated as long as there is a new image loaded and tested.

Then, bear in mind that the user always need to load a reference OCR image before testing. This can be done by clicking the button at the top right corner of the GUI. Please refer to my previous report for the reference OCR image.

Now the system is ready for the real testing tasks. The 'Testing' panel at the bottom of the GUI is designed to do so. After loading a new test image, a click at the 'Run' button can lead to a new resulting image and the corre-

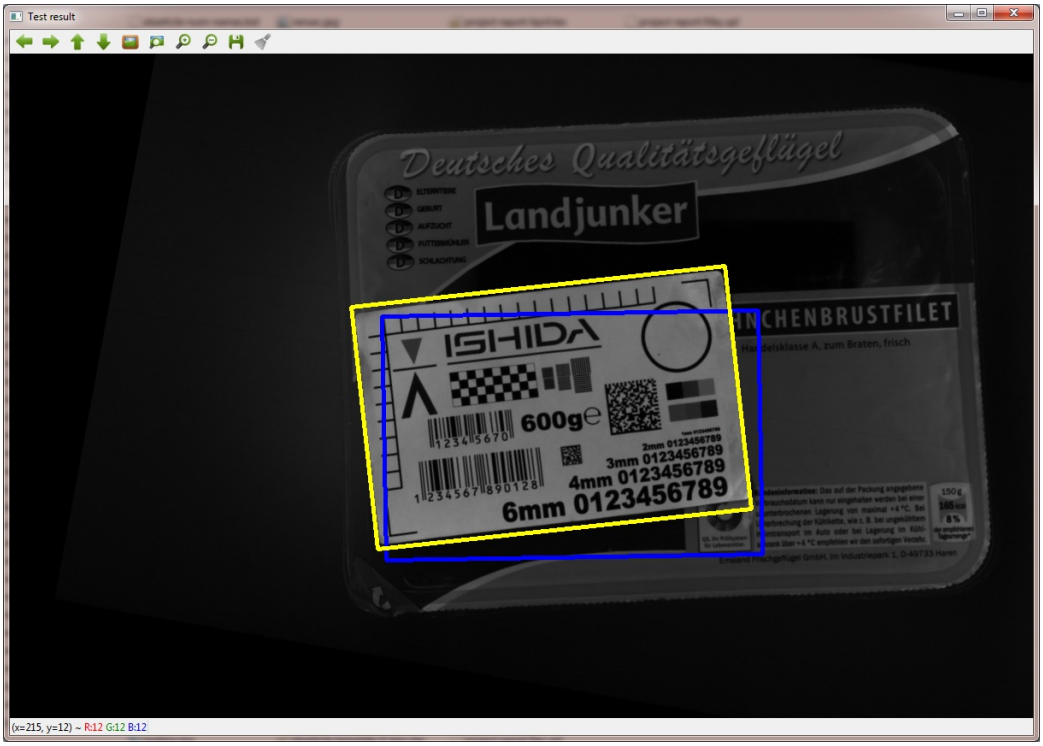

Figure 6: Test result

sponding statistics in the middle of the GUI will be updated. Figure. 6 shows a resulting image. If the position error is larger than 80 or the orientation error is greater than 30, a message box will pop out saying 'Should reject this tray!'. Otherwise it will show 'Accept this tray for OCR.' as shown in Figure. 7 and output the OCR result at the bottom right. In the example of Figure. 6, the final view of the GUI is shown in Figure. 8. Please note that the statistics in the middle have been updated.

3. Conclusions

In this report, we present a GUI for the label-TADD application software and explain how to use it properly.

Figure 7: Accept the tray for the OCR

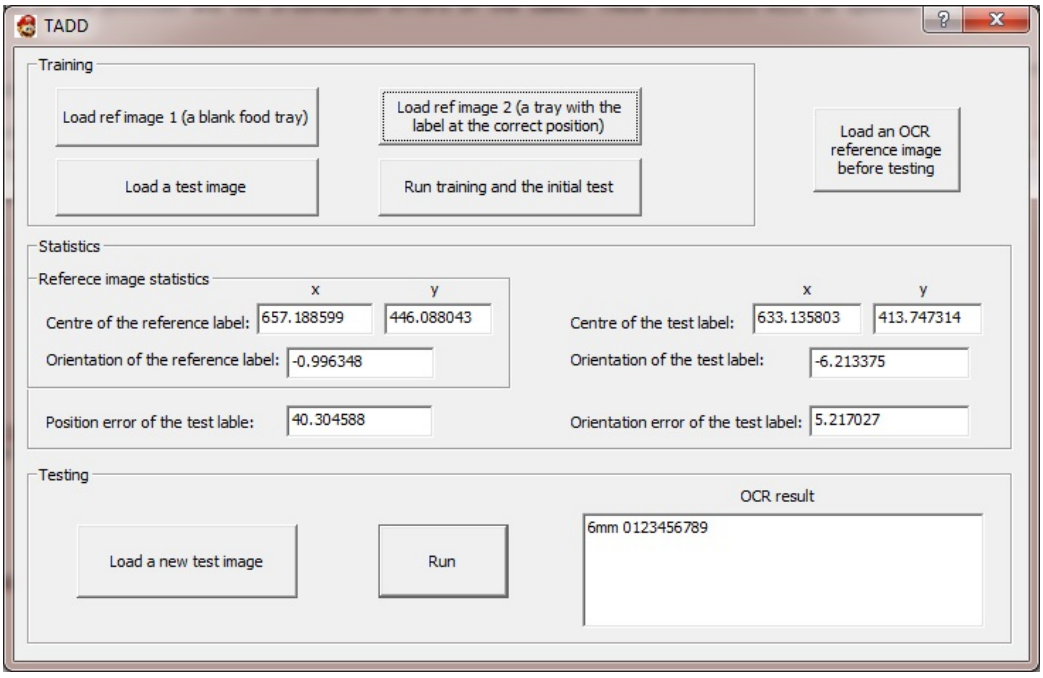

Figure 8: Updated statistics and OCR result

TSB-funded Project 'TADD' - Trainable vision-based anomaly detection and diagnosis Technical Report for June 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 June 2014, we evaluate the QA-TADD with respect to its capability of multi-label classification. The experiments are conducted in comparison with another state-of-the-art interactive learning and segmentation system—ilastik. Because the systems are interactive, a comprehensive comparison is very challenging. For instance, due to the interactivity involved in the QA-TADD, it is not fair to merely consider the accuracy of the classification result but how to evaluate the manual effort made by the user to train the system also matters. In this work, we try to deliver an evaluation based on roughly the same amount of the effort associated with the user interaction while quantitatively measuring the quality of the classification results, making it as comprehensive as possible.

1. Introduction

As pointed out by Kohli et al. [2], many successful applications of computer vision to image or video manipulation are interactive by nature. This is because problems in this field are known to be difficult, and very few fully automatic vision systems exist which have been shown to be accurate and robust under all sorts of challenging inputs. In the QA-TADD system, users are required to label training data, where such a process is significantly influenced by their prior knowledge and expertise, in order to help the computer to accurately classify various images. And the system is particularly designed to facilitate users to do so by using superpixels [1] to mitigate the manual

effort in this process. However, in our previous work (e.g., the BMVC submission), we treat the interactive QA-TADD system in the same manner as their fully automatic counterparts in the evaluation where the performance of the system was evaluated by computing the accuracy of the classification under some fixed set of training images. In this work, we report our latest evaluation of the QA-TADD system based on the real training data offered by users in an interactive manner. To measure the accuracy of classification, we compared it with another state-of-the-art interactive training and segmentation system. It is worth mentioning that, here we focus on the evaluation of multi-label classification, while in our previous work, we only consider a binary classification problem.

2. Competing system

The competing system used in this comparative evaluation is ilastik [3]. ilastik is a user-friendly tool for interactive learning and segmentation. Similar to QA-TADD, it has a convenient mouse interface for selecting training data manually. Their major differences are twofold. First, ilastik relies on pixelwise training data selected by drawing lines/brush strokes (although the width of the lines can be changed) in the image while QA-TADD asks users to select superpixelwise training data. Pixelwise and superpixelwise training data have quite different effects on the classification results and we shall demonstrate it later in this report. Second, ilastik trains a Random Forest classifier while QA-TADD employs an Adaboost classifier. We shall not discuss the technical details about the Random Forest method but just mention that it is also a widely used classifier for similar tasks in computer vision. And once the classifier has been trained on a representative subset of the data, it can be exported and used to automatically process a very large number of images.

3. Evaluation for multi-label classification

The evaluation is designed to answer the key question: how to measure the quantitative accuracy of the classification while considering the effort given by the users to train the classifier.

3.1. Training data

Certainly the quality of the classification depends on the training data. Usually the more and the more accurate the training data, the better the

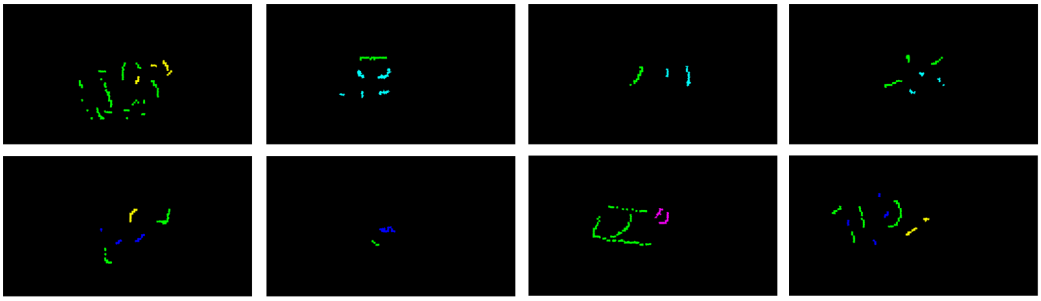

Figure 1: The training data for evaluating the QA-TADD

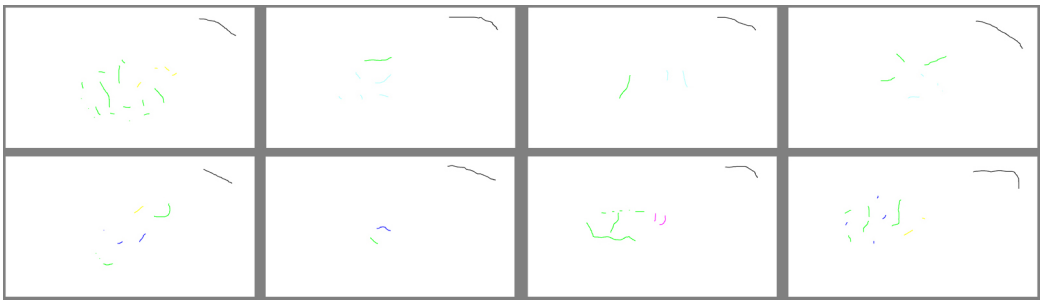

Figure 2: The training data for evaluating the ilastik (Please zoom it to 800% or more for seeing the strokes)

classification results. However, human users need to give more effort to generate a large amount of accurate training data. The difficulty of evaluating an interactive classification system is that besides its performance in terms of classification accuracy, the amount and the quality of training data input by users should also be considered and this is directly related to its efficiency in real applications. Therefore, to make a fair comparison, we try to generate two sets of training data which contain roughly the same amount of user effort with roughly the same accuracy. The set of training data used for evaluating the QA-TADD system is shown in Fig. 1 while Fig. 2 shows the training data that we used for the ilastik system. Note that since ilastik does not include an automatic background segmentation algorithm, we have to give some training data in the name of 'Label 0' to help the classifier recognise the black background. Because the background is completely consistent in any aspect, its segmentation is extremely easy for any classification method. Thus in this evaluation, we actually consider a 5-label classification task based on a set of images.

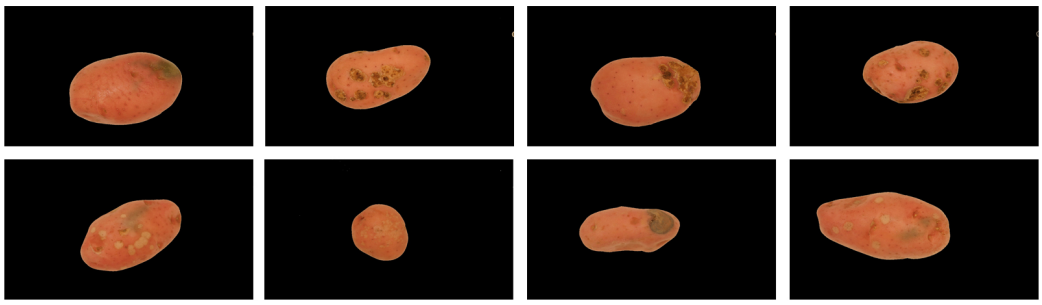

Figure 3: The original unmarked images corresponding to the training data

In Figs. 1 and 2, the different colours represent different potato defects or unblemished regions: Green—Unblemished region; Yellow—Greening; Blue—Black Dot; Cyan—Scab; Magneta—Other Blemish. It can be seen that the training data are composed of a set of strokes offered by the user in an interactive manner. These strokes are similar in terms of shape, position, and length, which ensures that the user gives roughly the same amount of effort in producing them. The only difference is that for the QA-TADD, the strokes are superpixel-wise while for the ilastik, they are pixel-wise. The original images corresponding to these training data are shown in Fig. 3.

3.2. Multi-label comparison

Fig. 4 displays some classification results produced by the QA-TADD with the correspondent ground truth images shown side by side. Note that the QA-TADD always tries to label every pixel in the image but there are a significant amount of pixels unmarked in the ground truth images. From the perspective of real quality control and grading application, it is difficult to grade the potatoes if quite a few pixels are unmarked. In this case, the QA-TADD tends to label the pixels unmarked in the ground truth images in green which means that they are unblemished. The reason behind it is that in the training data (the strokes shown in Fig. 1), most of them are labelled as green, which gives a large statistical variation of the appearance of the unblemished regions. As a result, the classifier is trained to have a high tolerance for the unblemished regions. And as you can see later in this report, this also partly explains why the unblemished label has a high recall rate.

In the quantitative comparison, we compute precision-recall maps for the classification results of each label using a set of 40 images. The ground

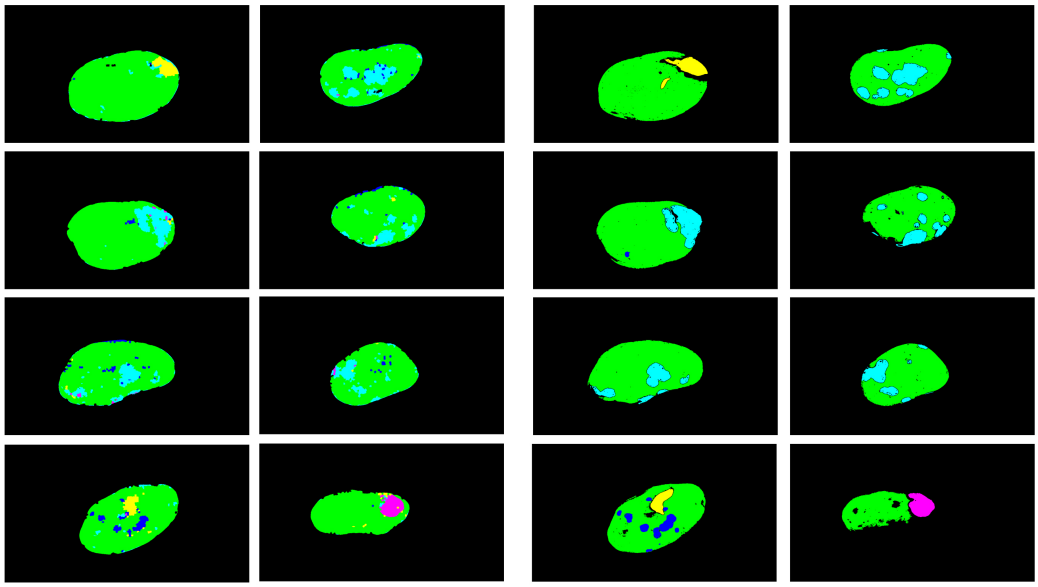

Figure 4: Labelling results of QA-TADD.

truth labelling of the 40 images are produced by a potato pathologist. The precision rate is computed as

$$precision = \frac{D_L \cap G_L}{D_L} \tag{1}$$

and the recall rate is computed as

$$recall = \frac{D_L \cap G_L}{G_L} \tag{2}$$

where D_L denotes the number of pixels assigned a certain label L by the classifier while G_L is the number of pixels with the label L in the ground truth images.

Fig. 5 show the comparative precision-recall maps of each label. Every point (red triangles or blue squares) appears in these maps are located by its coordinates expressed as $(precision, recall)$. Hence, if most of the points on a precision-recall curve are close to the top right corner $(1, 1)$, it means that the classification is good, and the closer the better. If a particular type of defect does not occur in some images, these images will not be taken into

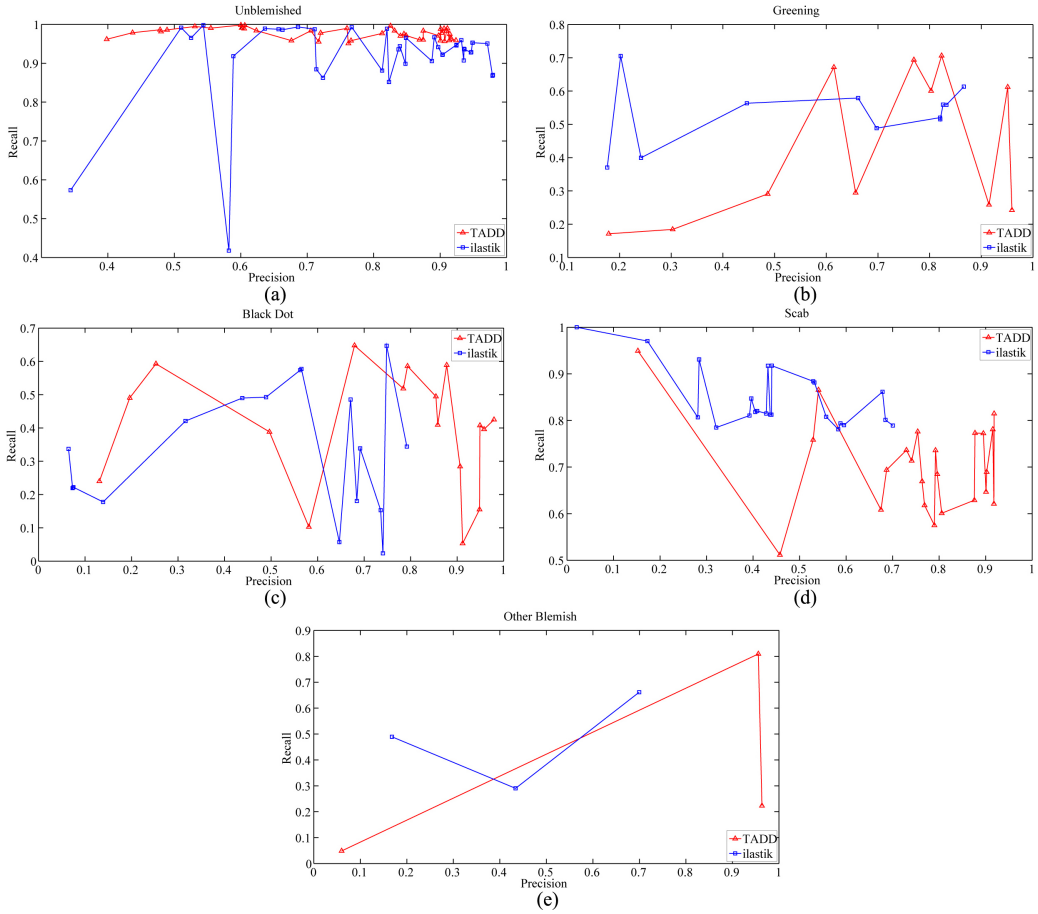

Figure 5: The precision-recall maps of the classification produced by QA-TADD and ilastik respectively. (a) Unblemished; (b) Greening; (c) Black Dot; (d) Scab; (e) Other Blemish

account when we draw the precision-recall map of the corresponding label. That is why the 'Unblemished' map (Fig. 5 (a)) contains the largest number of points (since every test image contains some unblemished regions). In general, it can be observed that the QA-TADD generates more points close to the top right corner in Fig. 5 although (e) is not quite meaningful since this type of defect only appears three times in our test images. According to Fig. 5, the more training date, the better the classification results. The 'Greening' (11 points in (b)) and the 'Other Blemish' (merely 3 points in (e)) defects are poorly classified because there are not enough training data offered to train the classifier to detect these two types of defects.

4. Conclusion

Based on the quantitative evaluations, we conclude that the QA-TADD outperforms the state-of-the-art system ilastik. Certainly, this conclusion is drawn based on a small set of test images and thus it is possibly not quite reliable. To deliver a more reliable evaluation, we need more test images and ground truth images. Ideally every type of defect should appear in at least 10 different images. In the QA-TADD, the training data are given in superpixels while the classification results shown in this report are presented in pixels. The evaluation thus demonstrates that superpixel-based interactive training and labelling system can deliver a good result although inevitably the transfer from superpixels to pixels introduces errors.

References

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

[2] P. Kohli, H. Nickisch, C. Rother, C. Rhemann, User-centric learning and evaluation of interactive segmentation systems, International journal of computer vision 100 (2012) 261–274.

[3] C. Sommer, C. Straehle, U. Koethe, F.A. Hamprecht, ilastik: Interactive learning and segmentation toolkit, in: Proc. 8th IEEE International Symposium on Biomedical Imaging (ISBI 2011).

TSB-funded Project 'TADD' - Trainable vision-based anomaly detection and diagnosis Technical Report for July 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 July 2014, I developed a simplified version of Label-TADD with GUI to make it compatible with the TADD desktop. Also, I set up the SICK Ranger camera and the corresponding software SICK Ranger studio with necessary configuration to output intensity images. In addition, I also investigated more general databases for evaluating the QA-TADD where most images are natural images.

1. Label-TADD updating

We have already developed a GUI (Graphical User Interface) for Label-TADD in May. However, we found that it cannot work properly in the TADD desktop in the back office of the Witham Wharf in Lincoln. After spending some time on it, we found that it is probably because the versions of the Qt (5.2.1) and the OpenCV (2.4.8) library installed in my own desktop and the TADD desktop are not quite compatible with each other. The latest version of OpenCV has incorporated some Qt functionalities into its framework. By calling some of these function, we can build up some fashionable control buttons when visualising an image as shown in Figure 1. Since the TADD desktop only has the older versions of Qt (4.7.3) and the OpenCV (2.3.1) library, the software which can run perfectly on my desktop crashes when running on it. Having made some effort on integrating the control buttons with the older versions of Qt and OpenCV, we found that the software still cannot work properly. Finally, to make it work with major functionalities

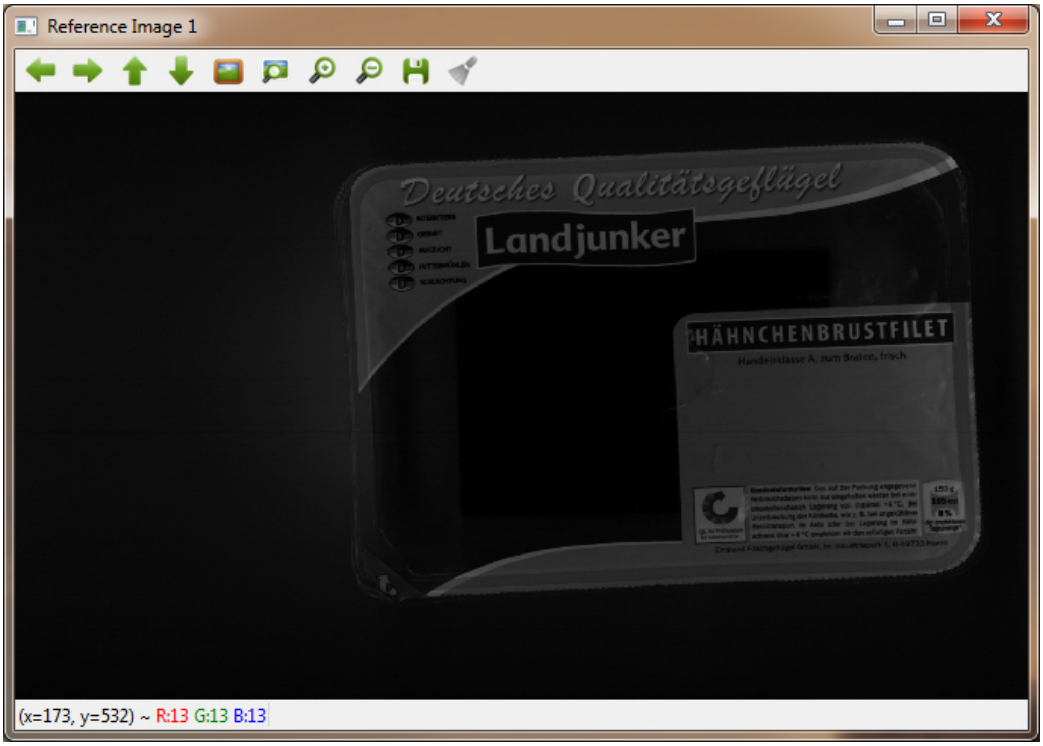

Figure 1: Image display with control buttons

(label detection and recognition, and OCR) on the TADD desktop for potential demonstration purpose, we decide to completely remove such image control buttons from the interface and just maintain a basic interface for the images when displaying them. The difference between the two versions of the Label-TADD is merely with or without the row of control buttons shown in Figure 1. Please note that the full version of Label-TADD can still work well on any computer (built up from the scratch) as long as we replace the DLLs and other linker files of the older version of Qt and OpenCV with their latest versions. We did not do so for the TADD desktop because the QA-TADD on it was developed based on the older versions and such a replacement could cause a crash of the QA-TADD.

2. 3D ranger camera configuration

We have already investigated the performance of the SICK ranger camera and configured it with the SICK Ranger Studio software. Since currently we

Figure 2: The intensity image taken by the SICK ranger camera

are still handling the safety issue of the Class 3b laser to meet the regulations of the University of Lincoln, we merely test its performance of capturing intensity images. A 20mm lens was mounted to the camera to take a photo for a fake potato under a normal lighting condition. As shown in Figure 2, the output is an intensity image with a resolution of 1536×512 in the BMP format.

3. General databases for QA-TADD

In July, we also spent some time looking for a suitable database for evaluating the QA-TADD system. Our target is to test the QA-TADD using some natural images while disabling the background segmentation function. The intention is to evaluate the interactive scheme of QA-TADD and the Adaboost classifier in a more general manner. Note that the evaluation of the quality of interactive segmentation algorithms is not straightforward. As the interaction in later steps typically depends of the segmentation result of earlier steps, there is no segmentation benchmark with identical behaviour for every segmentation algorithm evaluated. Furthermore, there are many different ways of interacting, starting from different scribble types over rectangles or ellipses up to boundary marking methods. Based on these considerations, many interactive segmentation methods have been presented without any comparable quantitative scores.

The most important segmentation benchmarks that have been applied to interactive segmentation are the GrabCut database (http://research.microsoft.com/en-us/um/cambridge/projects/visionimagevideoediting/segmentation/grabcut.htm) as well as the Berkeley Segmentation Bench-

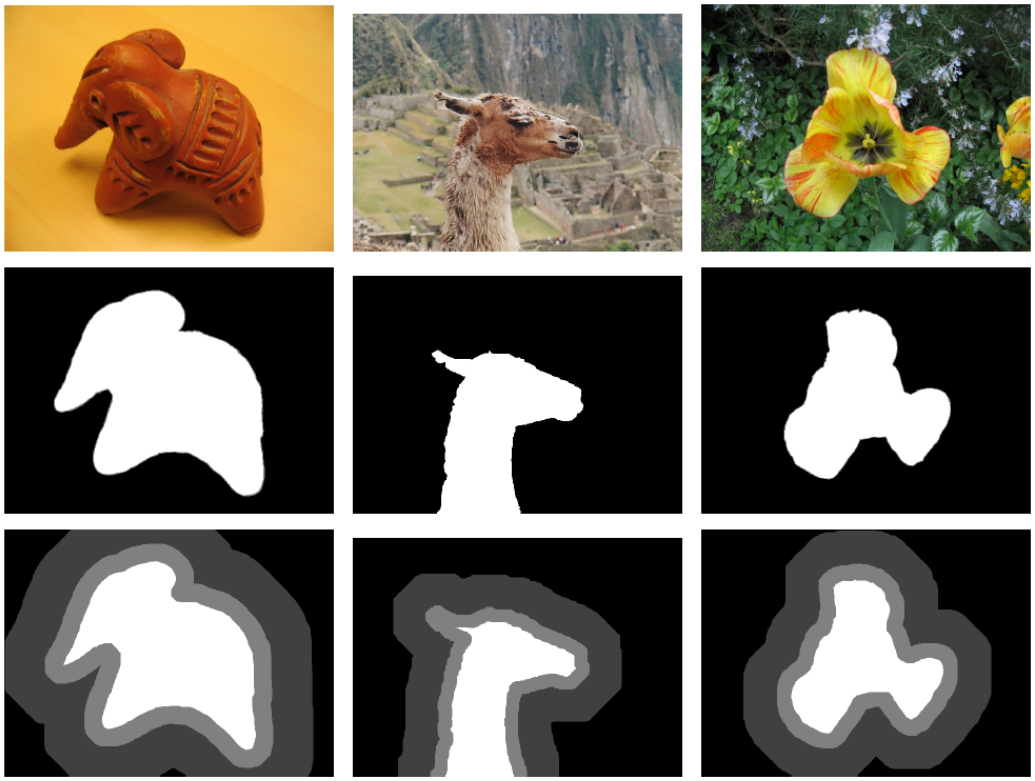

Figure 3: Three example images from the GrabCut database (first row) together with exemplary ground truth segmentations (second row). The third row shows the seeds for the segmentation algorithms: White pixels denote pixels for foreground model generation, dark gray pixels are used for background model generation. While black pixels are also background, they are not used for background model training. Light gray pixels are considered unknown and have to be assigned correctly by the segmentation algorithm.

mark (http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/). Other popular benchmarks used in image segmentation are the PASCAL VOC dataset (http://pascallin.ecs.soton.ac.uk/challenges/VOC/), the LabelMe dataset (http://labelme.csail.mit.edu/Release3.0/browserTools/php/dataset.php) and the IcgBench database (http://gpu4vision.icg.tugraz.at/index.php?content=downloads.php).

3.1. GrabCut Database

The GrabCut database has been widely used for for evaluating interactive segmentation algorithms The database consists of 50 images, where 20 images

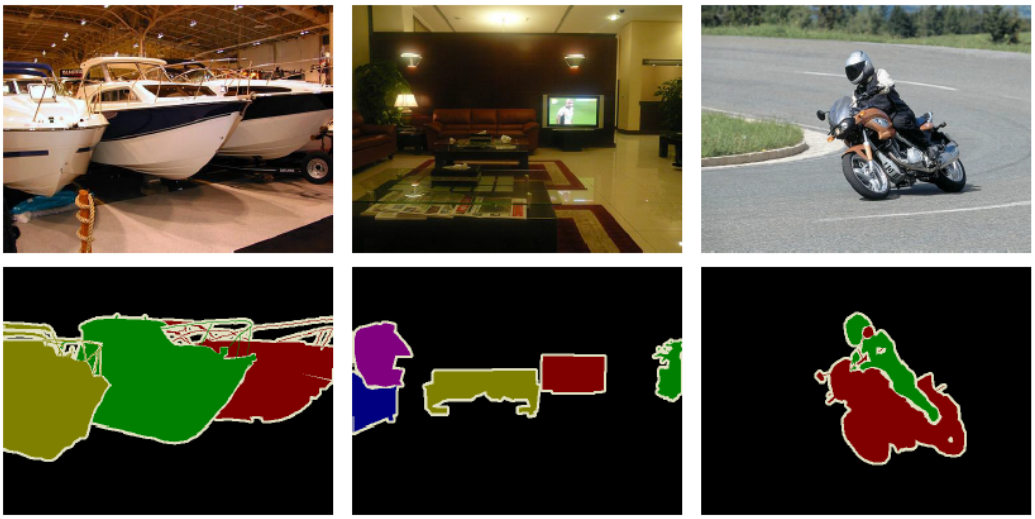

Figure 4: Exemplary images of the PASCAL VOC segmentation dataset together with groundtruth segmentations.

are taken from the BSDS300 dataset (see below). For every image, a ground truth segmentation is given as well as a rough labelling based on the object boundary, from which seed pixels are deducted. Figure 3 gives an example for the GrabCut database.

3.2. Berkeley Segmentation Dataset and Benchmark

The Berkeley Segmentation Dataset and Benchmark was initially presented as BSDS300, consisting of 300 color images. Recently, the dataset was extended by 200 images to form the BSDS500 database. For every image in the dataset, there exist multiple ground truth segmentations manually drawn by different users.

3.3. PASCAL VOC Database

The PASCAL Visual Object Classes (VOC) Challenge is a popular object recognition competition on a large database, which is extended every year. Since 2007, the challenge includes a segmentation benchmark, where the task is to assign every image pixel one out of 20 classes: aeroplane, bicycle, bird, boat, bottle, bus, car, cat, chair, cow, dining table, dog, horse, motorbike, person, potted plant, sheep, sofa, train, and TV/monitor. The ground truth segmentations provide a pixel-wise labelling into either one of these 20 classes, a background class and an unlabelled class (see Figure 4).

3.4. LabelMe Database

The LabelMe database consists of images labelled by arbitrary people via a web annotation tool. During annotation, the users have to identify objects within images, loosely draw their outline with polygons and freely choose a descriptive name for the object (e.g. car, building, tree, person, window, road, sidewalk, sign, sky etc.). Groundtruth segmentations are also offered in the database.

3.5. IcgBench database

In order to evaluate interactive image segmentation techniques thoroughly, the IcgBench database benchmark was created by gathering many images of varying objects, people, animals and scenes. Seed pixels as well as ground truth segmentations corresponding to the interpretation of the images are also provided by different people. The benchmark exhibits segmentations with several regions in a single image. It can also be reduced to evaluating foreground/background segmentations by translating multiple labels to sequential binary problems. The seed pixels are given by scribbles drawn into the region to segment rather than by geometric primitives such as ellipses or rectangles approximating the object boundary. Figure 5 gives example images of the IcgBench database.

It seems that the IcgBench database is very suitable for evaluating the QA-TADD system since it is designed for benchmarking multi-label segmentation methods and also offers a good way to simulate potential user interactions. So at the current stage, we plan to use this database to evaluate the QA-TADD and compare it with the state-of-the-art techniques.

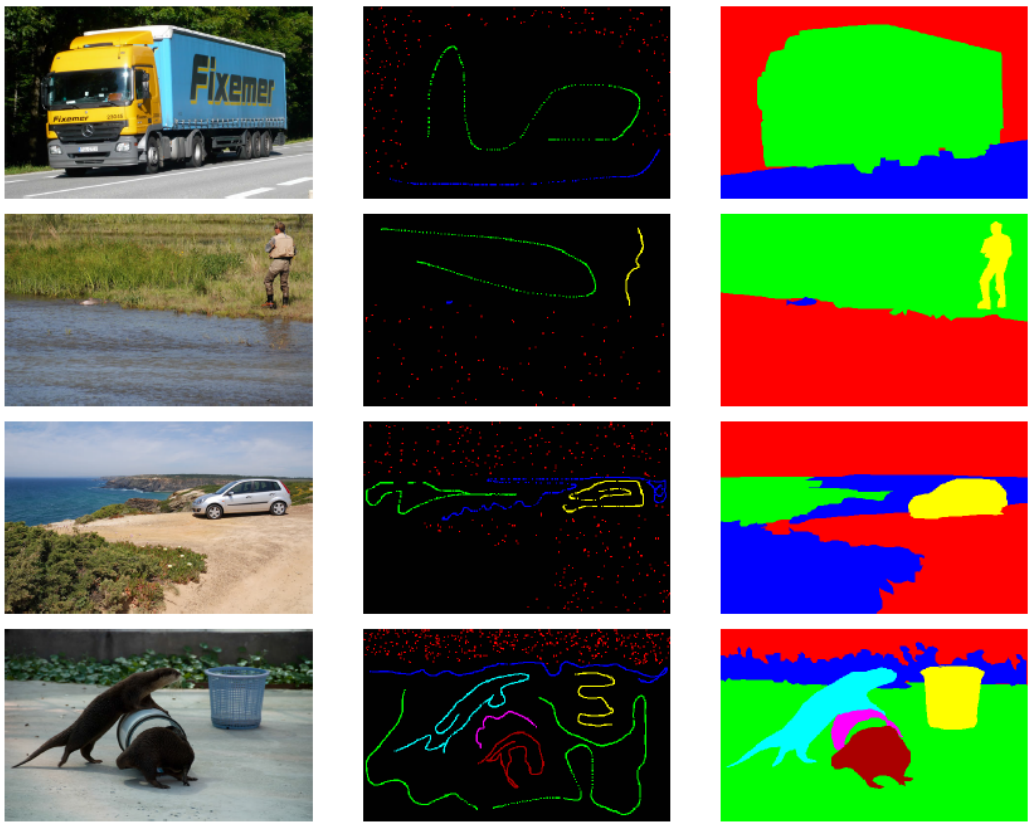

Figure 5: Exemplary seed—groundtruth pairs of the dataset IcgBench. Note that the images showing seed pixels (centre column) have been dilated for better visibility.

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

Hossein Malekmohamadi and Tom Duckett

Chapter 1

Introduction

This is the May 2014 report for the project TADD. The core topic of this report is software development for QC. The job description for this month is taken from objectives of QA-TADD for Q6:

1. Save function/Editable training sessions: Extend to multiple images per classifier. We have tested multiple-image classifier for both interactive training and ground truth training and each has more than two type of blemishes.

2. Finalise universal TADD, where functionality of the two previous systems (the original one developed by Jame Hutton for his MSc thesis and the version at Sutton Bridge amended by Michael Barnes) to provide a single unified system. From now on, we use QA-TADD to refer to this version of TADD developed in this TSB-funded project.

Chapter 2

QA-TADD Development

In this section, first we give details on TADD software developments consisting of editable training sessions for many images. In the last quarter, we implemented editing ability for saved classifiers but that one was only for one image per classifier. The reason for many images per classifier was the memory leak that was leading the system to crash. This is now fixed by implementing new functionalities for QA-TADD that includes: LoadData(), NextAction(), and PreviousAction().

Editing multiple-image classifier gives the ability to select some images from a saved classifier, edit mark ups and finally the user can save it as a new classifier. We have created another function to train a multiple-image classifier with all images at once after it is loaded which is called: trainAllAction(). Afterwards, based on the ground truth data, we have created classifiers containing several images and trained TADD and tested it with new images Figures 2.1 to 2.4.

Furthermore, we have updated QA-TADD with these new features and uploaded to the project Github repository. All Figures 2.1 to 2.4 are performed with the latest QA-TADD.

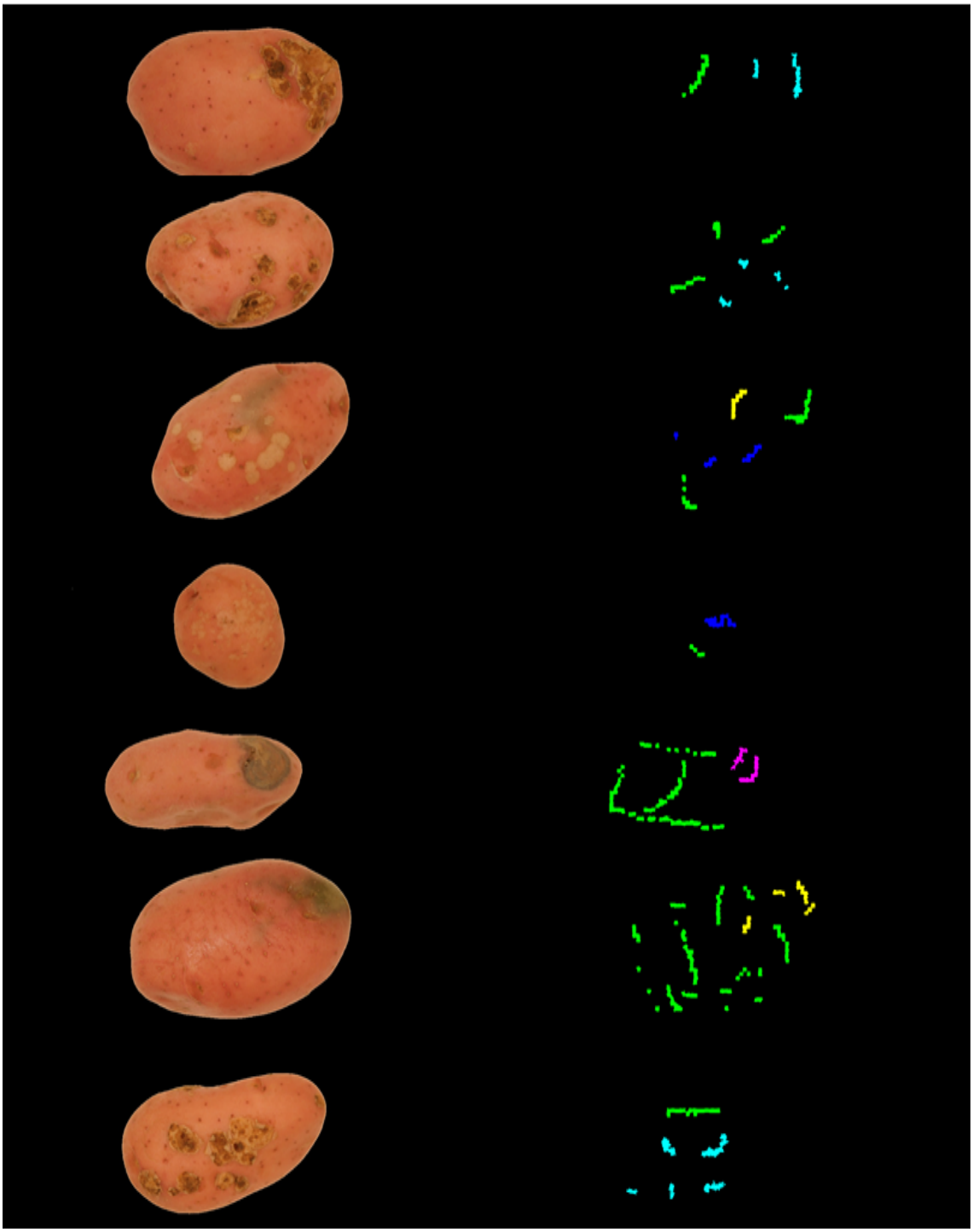

Figure 2.1: Schematic of a saved classifier which is later loaded and trained with all its images. Left column is the unmarked images and right column is the mark up locations done interactively.

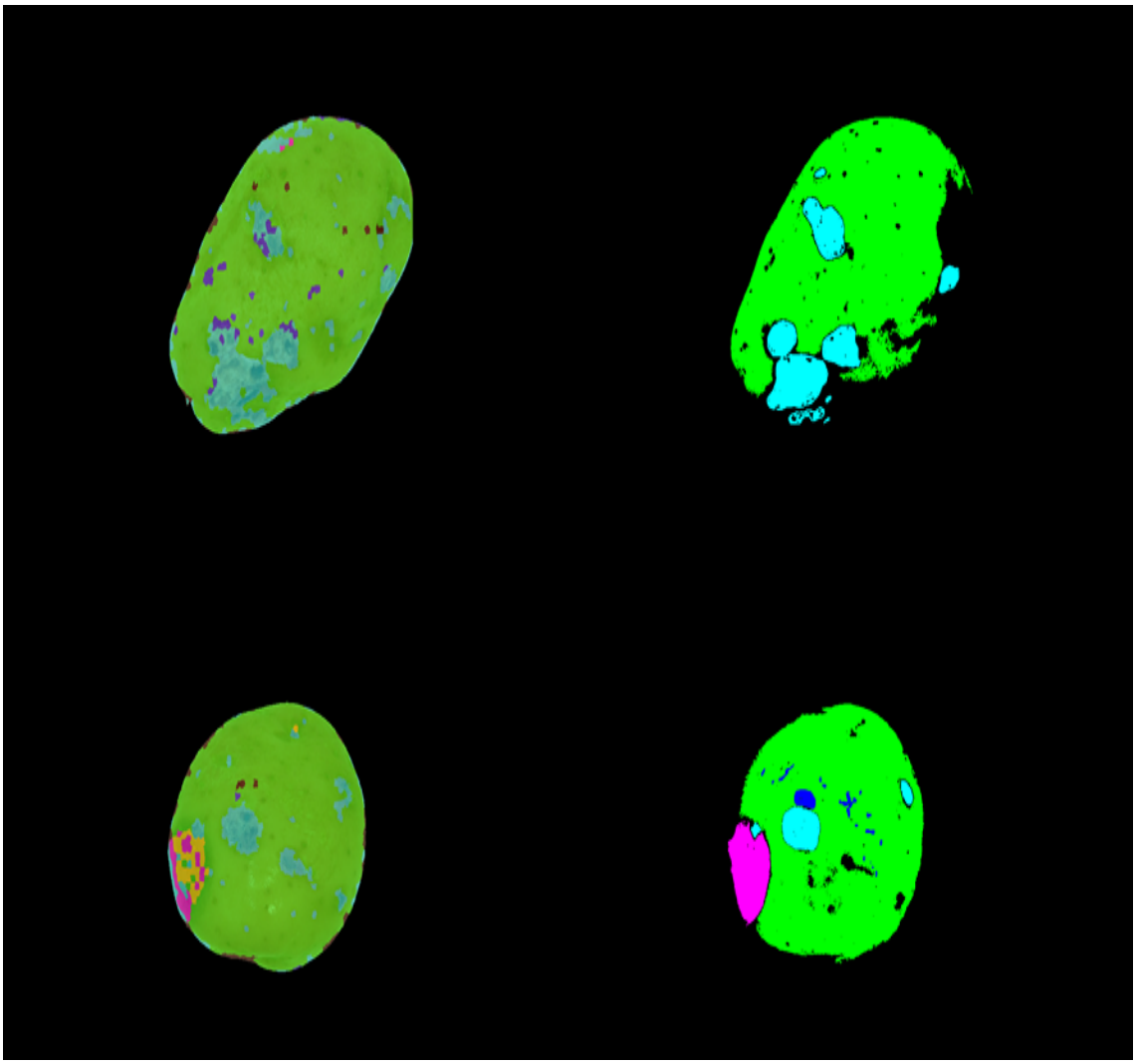

Figure 2.2: Test results for two sample images trained with a saved classifier containing 7 images (interactive training). Left column depicts the results and we compare them to the ground truth data in the right column.

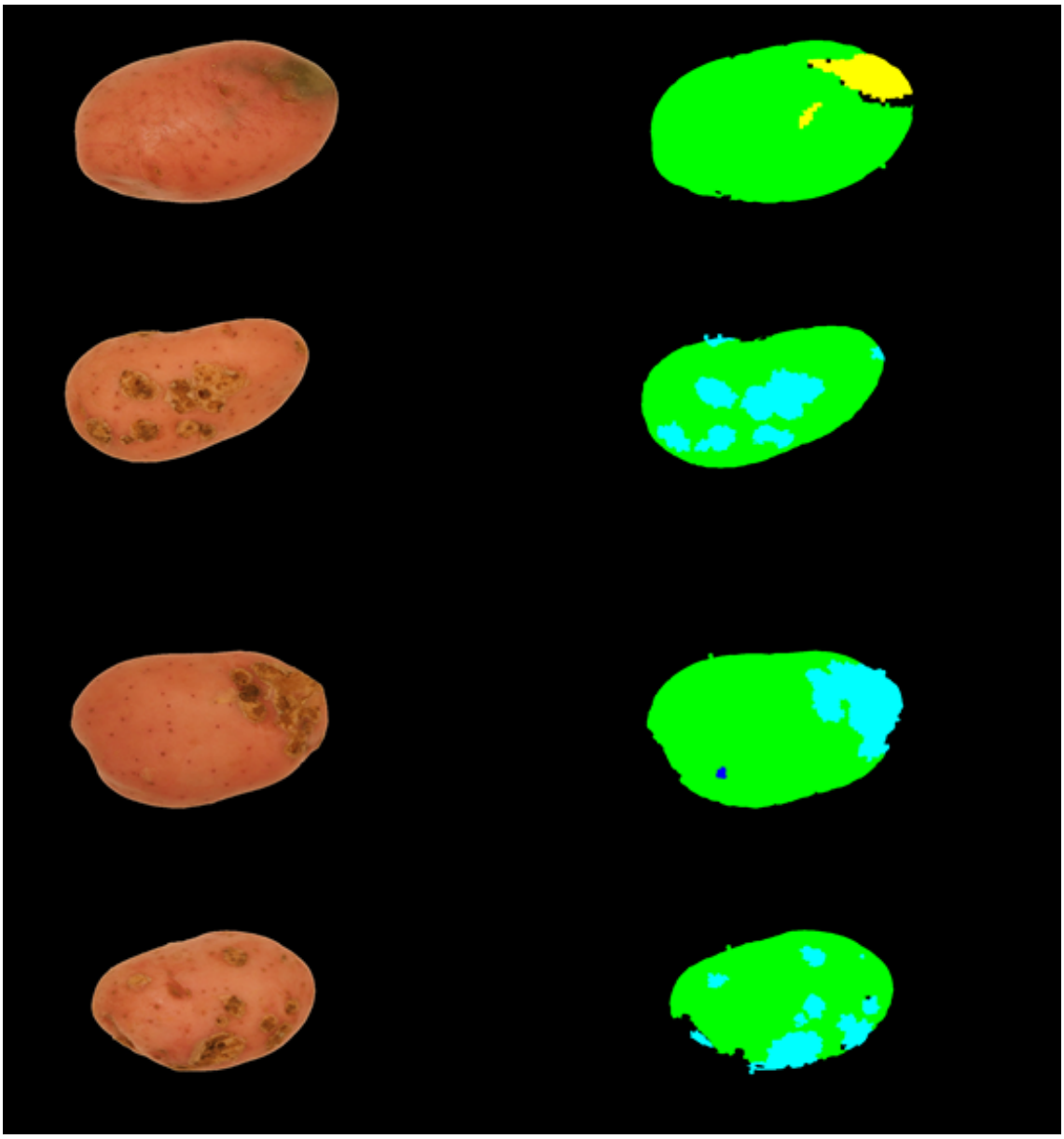

Figure 2.3: Schematic of a saved classifier which is later loaded and trained with all its images. Left column is the unmarked images and right column is the mark up locations converted directly from ground truth data.

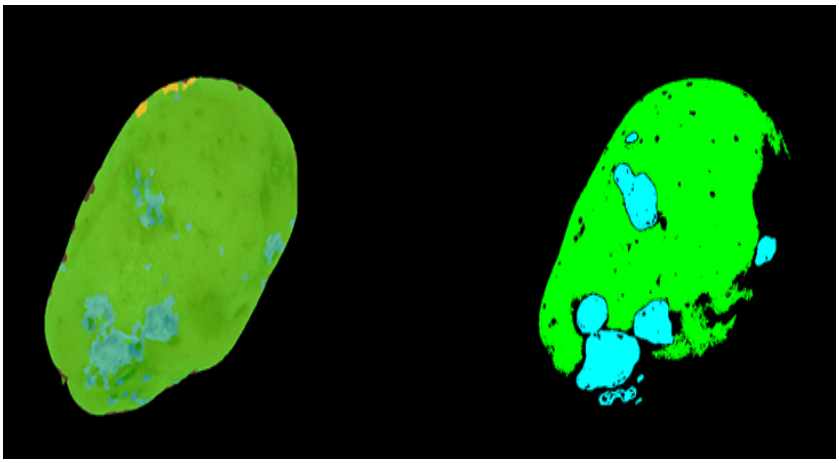

Figure 2.4: Test results for one sample images trained with a saved classifier containing 4 images (ground truth data). Left column depicts the results and we compare them to the ground truth data in the right column.

Chapter 3

Conclusions

In this month (May 2014), we had quarterly review meeting on the first week. Afterwards, main developments for QA-TADD achieved in this month are: 1) save, load, edit, train classifiers with many images interactively and off-line (trained with ground truth data) with more than two blemish types, and 2) finalising the QA-TADD.

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

Hossein Malekmohamadi and Tom Duckett

Chapter 1

Introduction

This is the June 2014 report for the project TADD. The core topic of this report is software development for QC. I will be away on holidays from 13 to 30 June 2014, consequently this results covers research and development in QA-TADD for the first two weeks of June. The job description for this month is taken from objectives of QA-TADD for Q6 which is to implement dual resolution in TADD to be fed with high resolution images from DSLR. These developments are for QA-TADD which was finalised in the last month (May 2014).

Chapter 2

QA-TADD Development

In this month (June 2014), we achieved the goal to enable QA-TADD work with high resolution images. Previous resolution was 1280×720 and now it is 3072×2048 which is 583% increase. To avoid gslic segmentation algorithm saturation for high resolution images, the number of superpixels has been reduced to 5000 from 9200 which is 46% decrease. This will increase each superpixel size accordingly for higher resolution images and does not affect performance in our case. Saturation means that after a certain number of iterations in gslic, segmentation will not proceed and all segments in right hand side of the image will be the initial blocks as shown in Figure 2.1. In Figure 2.2, we can see the same image shown in Figure 2.1 but with fewer superpixels. Working with high resolution images is accomplished by modifying memory allocations and decreasing the number of superpixels without affecting TADD performance. All other functions including saving, loading and editing classifiers (no matter how many images they have) are working accordingly. An example of training and classifying with one image is shown in Figures 2.3 and 2.4.

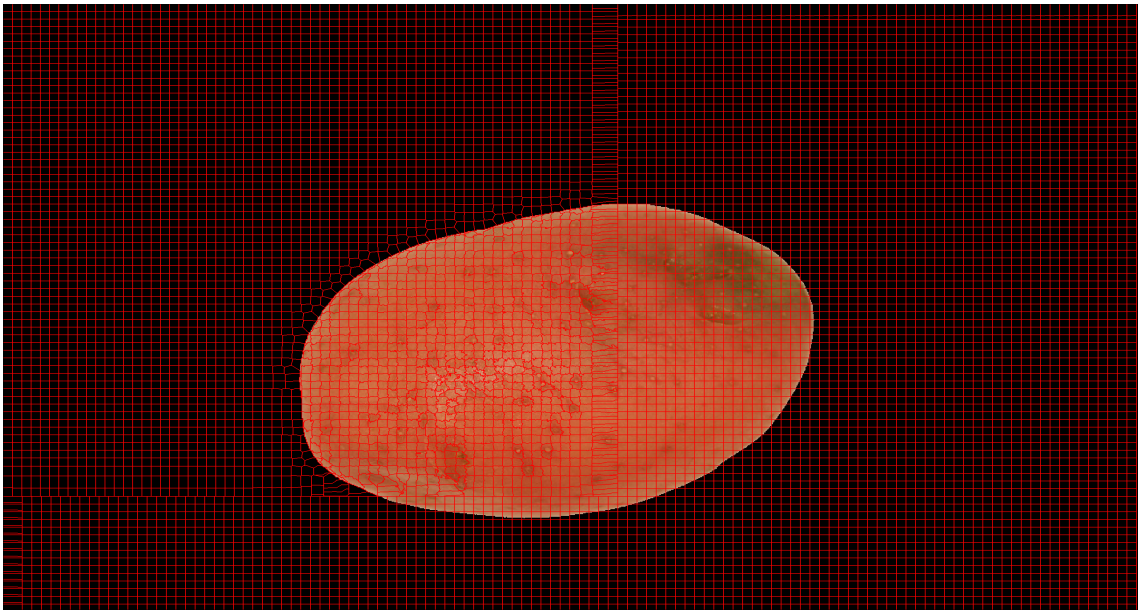

Figure 2.1: Gslic segmentation for high resolution image with 9200 super pixels. The algorithm will not proceed after a certain number of iterations as it is visible in the middle of the image.

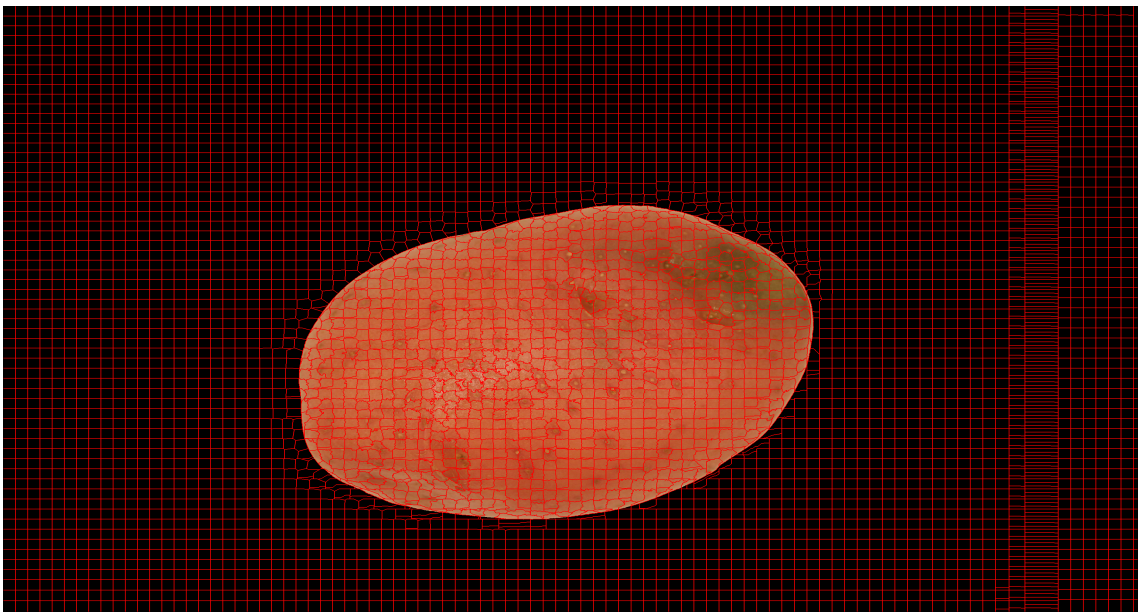

Figure 2.2: Gslic segmentation for high resolution image with 5200 super pixels. Saturation point have moved to the right compared to Figure 2.1.

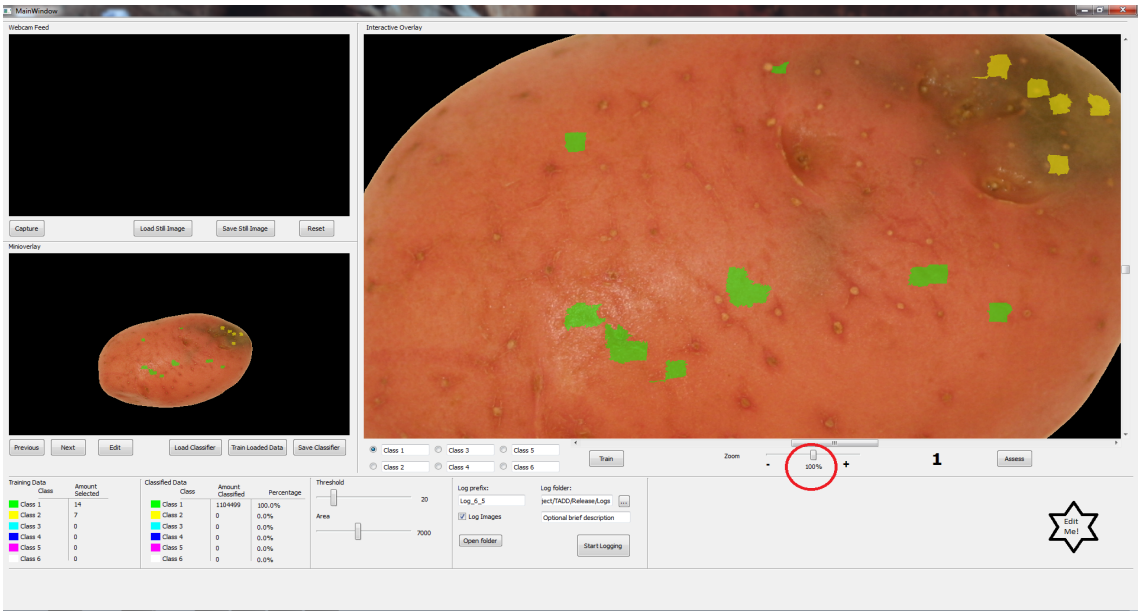

Figure 2.3: Selecting data for training from a loaded high resolution: image is shown in its original size (100% zooming is on which is marked up in red in the picture).

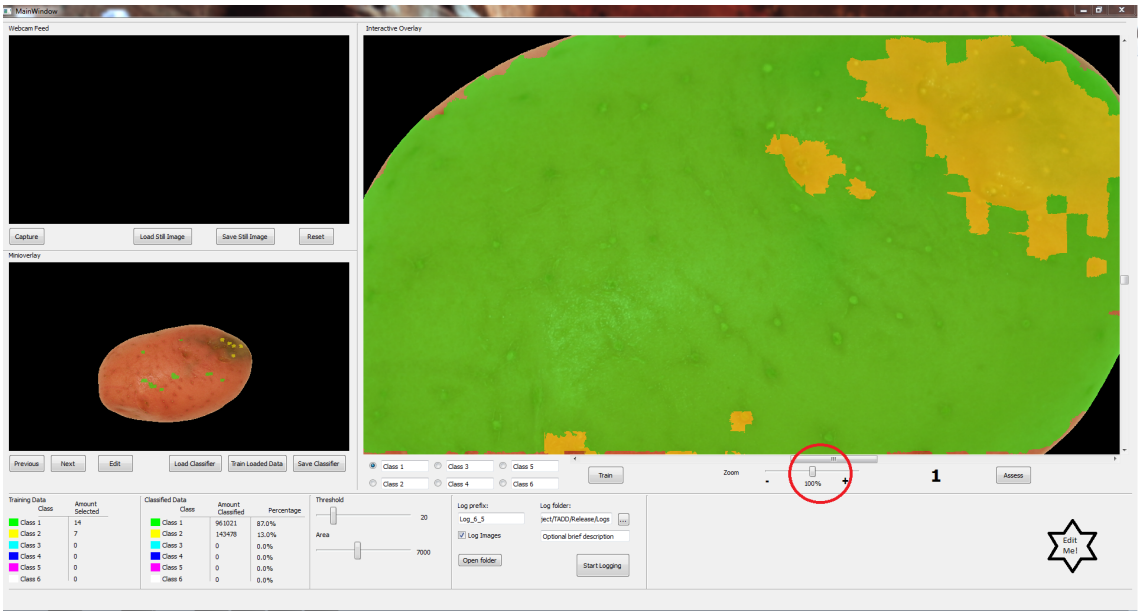

Figure 2.4: Classifier results: image is shown in its original size (100% zooming is on which is marked up in red in the picture).

Chapter 3

Conclusions

In this month (June 2014), we have enabled TADD work properly with high resolution images.

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

Hossein Malekmohamadi and Tom Duckett

Chapter 1

Introduction

This is the July 2014 report for the project TADD. The core topic of this report is software development for QA-TADD. I will be away on training course from 21 to 25 July 2014, consequently this report covers research and development in QA-TADD for the first two weeks and the last week of July. The job description for this month is taken from the objectives of QA-TADD for Q6 which is integrating the purchased camera into the QA-TADD to take high resolution images.

Chapter 2

QA-TADD Development

In the last month (June 2014), we achieved the goal to enable QA-TADD work with high resolution images. To enable QA-TADD users work with real-time images, we purchased a decent DSLR camera (Canon EOS 700D Digital SLR Camera with 18MP CMOS sensor and 3 inch LCD) with the prime lens (Canon EF 50mm - f/1.4 USM Lens). A prime lens is a lens that has only one focal length. Prime lens can provide crisp and sharp shots. The effective distance in order to capture the container box (60 cm $\times$ 40 cm $\times$ 20 cm) fully is around 87 cm. In addition, we purchased a UV filter for the DSLR camera that absorbs the ultraviolet rays to make images under blueish or yellowish LED lights look more natural. Software implementation for this part of QA-TADD is to integrate DigiCamControl[1] in to it as shown in Figures 2.1 to 2.4.

[1] http://digicamcontrol.com/

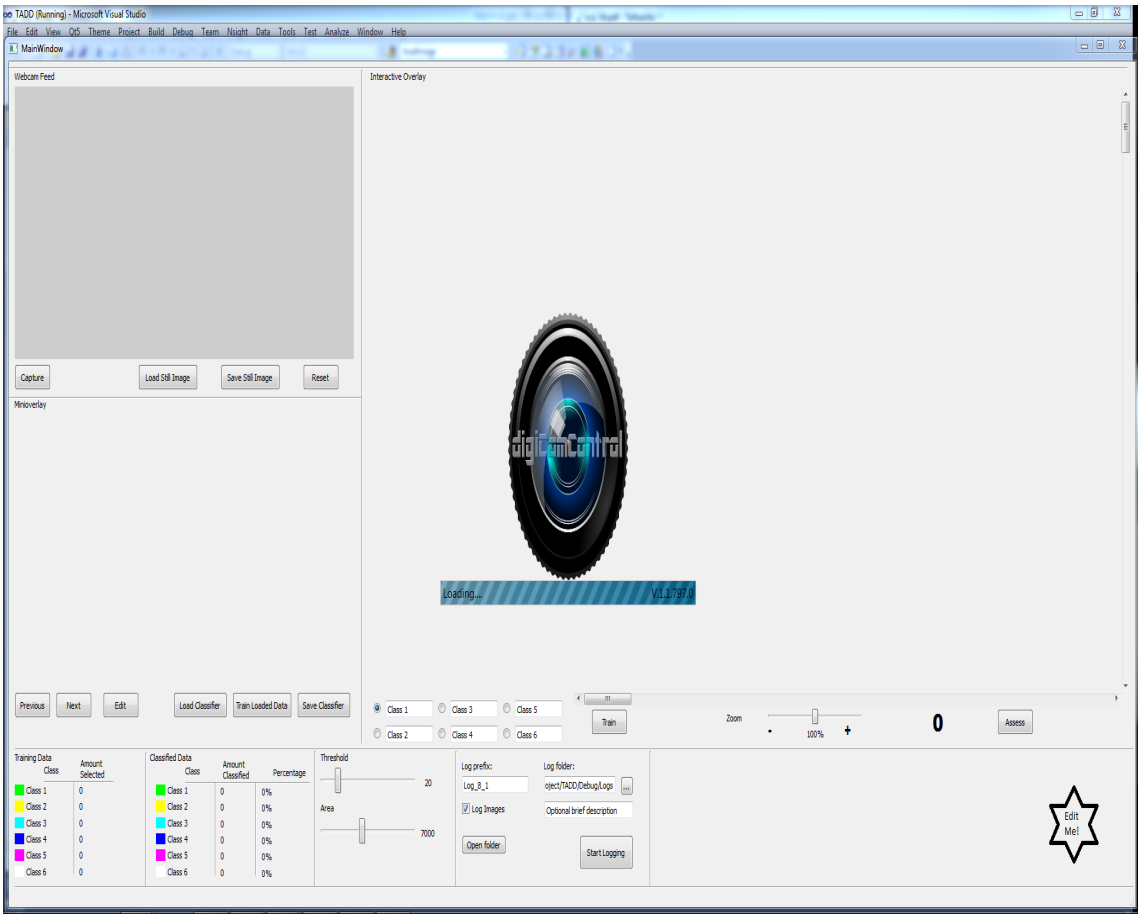

Figure 2.1: QA-TADD is running while it calls integrated DigiCamControl UI.

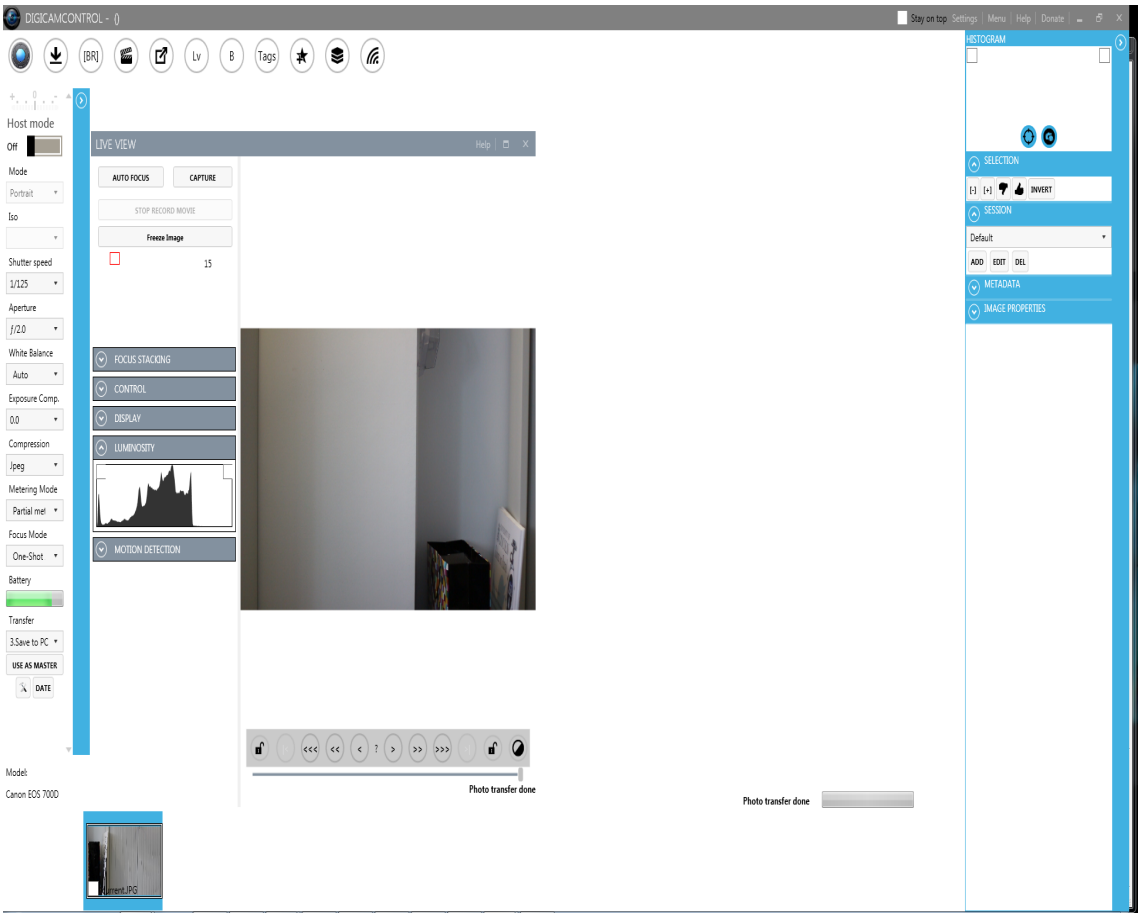

Figure 2.2: DigiCamControl UI is shown with functionalities like live view and capture. User can choose different parameters as well like shutter speed, aperture, auto-focus and white balance.

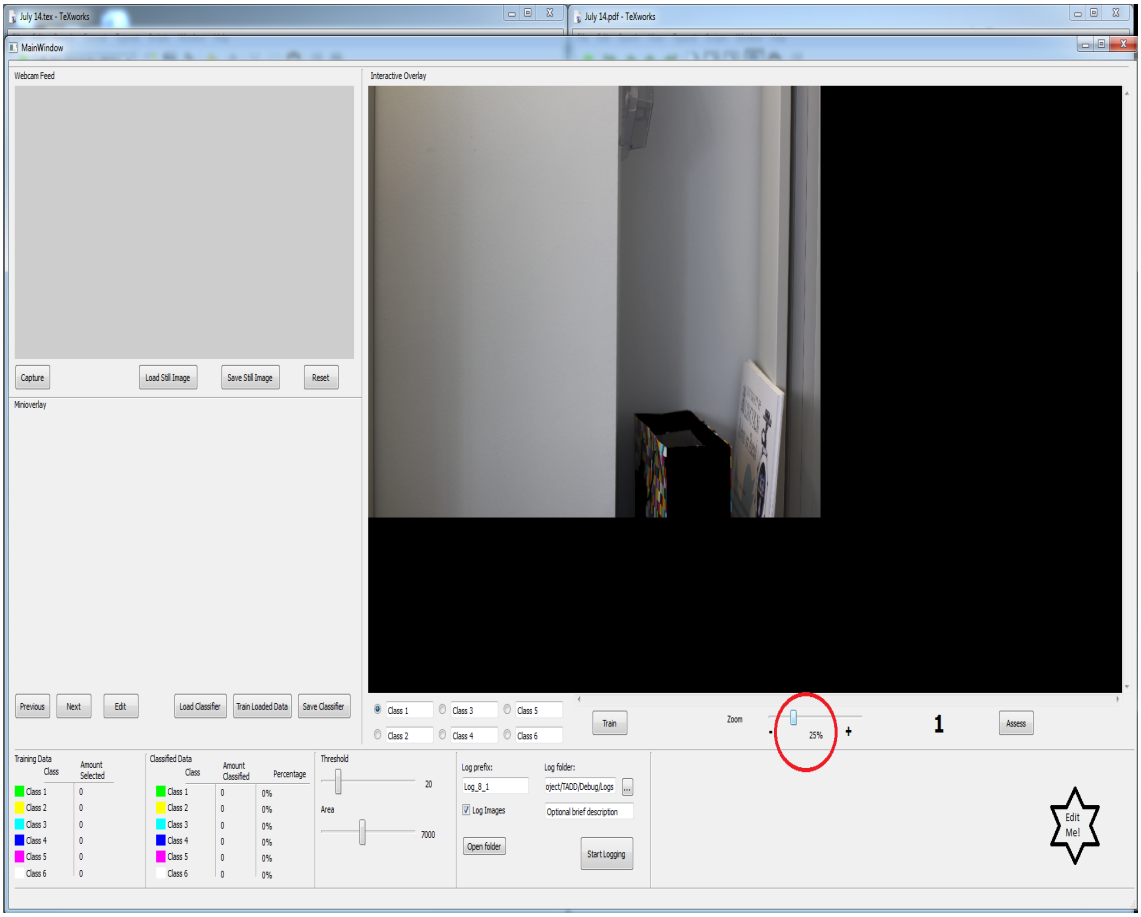

Figure 2.3: QA-TADD is then loaded with the captured photo with DSLR. Please note that the zoom is set to 25% for viewing purposes.

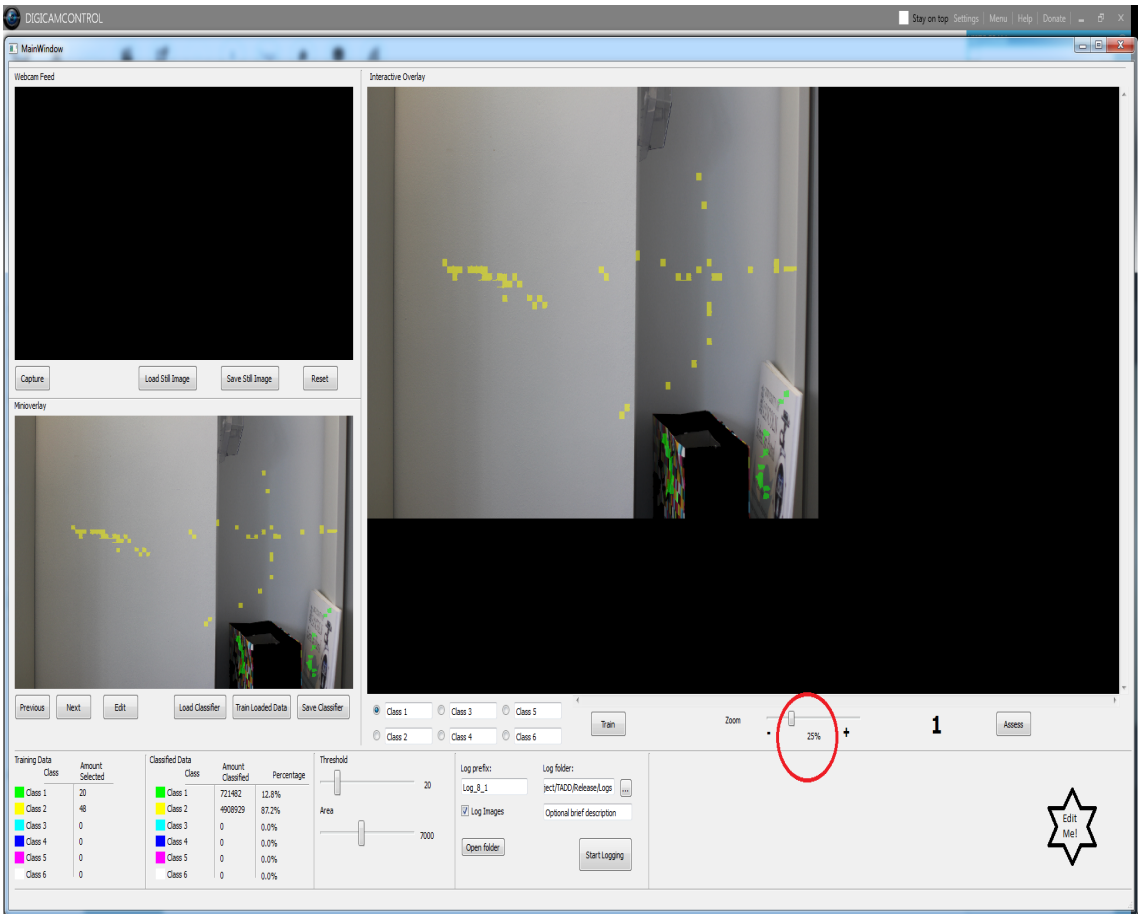

Figure 2.4: Training process is shown for the captured image.

Chapter 3

Conclusions

In this July 2014 report, we have enabled QA-TADD to work with the purchased DSLR camera. This gives the ability to the user to process QC tasks with more detailed images form potatoes.