Ročník 2011 electroscope Číslo II

DISPLAY INSPECTION SYSTEM

T. Babinec [1], P. Cip[1]
[1] Department of Control and Instrumentation, Faculty of Electrical Engineering and Communication, VUT, Brno, Kolejni 2906/4, Brno
E-mail : xbabin01@stud.feec.vutbr.cz, xcippa00@stud.feec.vutbr.cz

Abstract:
Graphical user interfaces are often the best solution for creating an easy to use human machine interface. Their development has an iterative character and requires periodical testing of the functionality and consistency of the graphical output. This paper deals with the design of a semi-automated system for functionality and quality inspection of graphical user interfaces in various devices and screen types. The presented approach is based on image processing algorithms, which minimizes the need for human interaction during the test procedures.

INTRODUCTION

One of the essential characteristics of a modern human machine interface (HMI) is the ability to communicate with user in some graphical form. This is due to the importance of visual perception among other human senses.
It is a well know fact, that images have the ability to express large amount of information in a very efficient and easy to understand way. As a consequence various sorts of electronic devices (e.g. PDAs, cell phones, GPS, etc.) utilize one or more graphical screens which can be based on a broad variety of different technologies. Several the most widespread technologies are the black&white fixed segment and dot matrix displays, which are typical for simple or low-end devices. Another important group includes full-colour LCD (liquid crystal displays), TFT (thin-film transistor) and OLED (organic light emitting diode) displays. An example of an unconventional but rather interesting screen technology is the E-INK (electronic ink), which significance and popularity among users has been growing rapidly in the recent years.
The development and mass production of nowadays complex electronic devices requires extensive functionality and quality inspection. A very important component of the inspection is the HMI evaluation, which can be divided to hardware (HW) and software (SW) oriented approaches. An example of inspection system oriented on the detection of faulty HW features is described in [1].
Completely different category of inspection mechanism is required by end-user device manufacturers and graphical user interface (GUI) developers. Their never ending competition for better looking, more user friendly and intelligent software environments leads to extremely complex screen content. The content is usually compiled from sophisticated menu structures and other graphical objects, which change dynamically according to actual situation and user input.
Generally such GUI development requires repeated fine adjustments to the SW structure, which may cause unexpected behavioral and random errors in the whole user interface. Therefore a rather extensive check of GUI consistence and functionality is required after every significant change in the SW.
The test procedures are monotonous and time consuming. As long as they are performed by human operators, the test results can be affected by operator's fatigue or other influences that are difficult to evaluate. In order to minimize or completely replace human interaction during the tests, an automated inspection system has to be designed. In [2] a method for automatic GUI inspection based on a source code analysis has been presented.
This paper describes the design of a semiautomatic inspection system based on computer vision algorithms. The system enables analysis of already displayed graphical information on various types of screen devices.

SYSTEM DESIGN

In order to satisfy the development goals, which are presented in the next subsection, a complete inspection system including HW and SW components had to be designed. The development was focused at imitating humanlike behaviour during the testing process. Therefore the system facilitates visual evaluation together with the possibility of on-line human input simulation as a reaction to currently displayed information.

Requirements

- Maximum separation between the inspection system and tested devices, which minimizes mutual influence and error propagation.
- A simple way for rapid test procedure scripting, provided by well organized programming interface.
- An uncomplicated functionality expansion.
- Minimum need for human interaction.

Hardware set-up

As it is depicted in figure 1, the HW set-up of the inspection system can be assembled from ordinary components according to the specific needs of the tested device. Basic HW element is an Imaging Source industrial camera connected to a PC. In order to be able to process colour information with a pixel resolution sufficient for majority of various display HMIs a 41BU02 camera model equipped with 1280x960 RGB CCD sensor with Bayer encoding capable of 15 frames per second was used.
An adjustable stand ensures proper camera positioning, which has to be adapted to available optics and dimensions of the tested screen. For stabilizing the lighting conditions or passive screen (e.g. E-INK) inspection an additional light source may be required.
Image processing and potential device control over some input simulator module is accomplished by the inspection SW installed on the PC workstation.

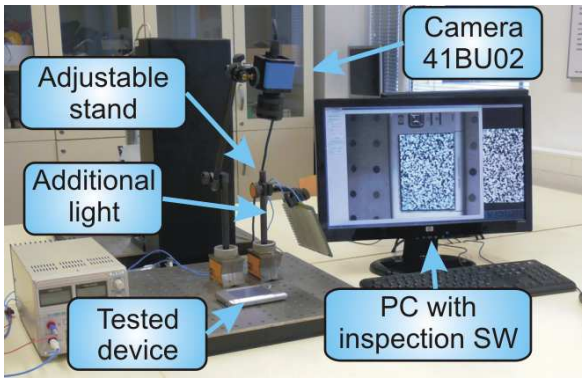

Fig. 1: Inspection system hardware set-up

Software environment

The implemented inspection SW has a modular structure (see figure 2). For initial system calibration and experiment set-up a “Screen Inspector SetUp” module was created. Its purpose and functionality is described more closely in the following section of this paper. The communication with camera HW, properties management and image grabbing is implemented in the module “UsbCam”. To enable online and automatic control of the inspected device a custom built functionality represented by “Input Simulator” module may be linked to the system.
The image processing core of the software environment is hidden inside the three-level module. The implemented computer vision functions are based on the programming resources available from OpenCV library [3]. High performance computer vision algorithms are written in C/C++ language and define the LowLevel part of the image processing core. This functionality is wrapped inside the HighLevel object hierarchy, which was created using CLR C++ and is intended for .NET managed environment. At the same time HighLevel objects and methods represent a sort of a programming interface, which allows fast and uncomplicated test procedure scripting with full-featured programming languages like C# and therefore satisfies one of the development goals.

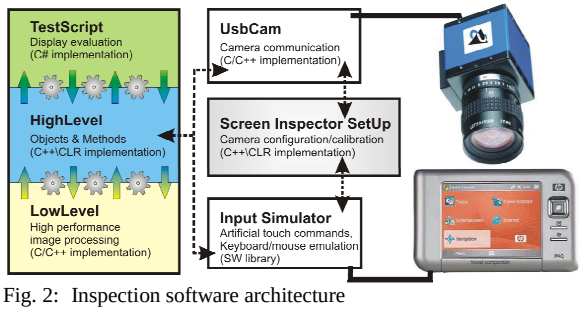

Fig. 2: Inspection software architecture

OPERATION & RESULTS

In order to ensure better results and deterministic behavioural of the designed inspection system, a system calibration is required before initiating the actual test procedure. Implemented calibration procedure, inspection functionality, achieved results and proposed scheme for GUI content definition are discussed in detail in the following subsections.

Fig. 3: Screen template (top) and its mask (bottom)

Display description

The expected GUI content is described using a XML document, which is an essential part of every test procedure. The XML structure is designed to achieve the most flexible but at the same time simple way for

definition of hierarchical composition of graphical objects.
The objects tested like screens, buttons, etc. are divided into classes with specific behavioral, which is described in the test script. Each basic screen element has unique features described as a collection of ideal bitmap templates (e.g. figure 3 top) combined with detection masks (e.g. figure 3 bottom) and rectangle coordinates defining areas intended for optical character recognition (OCR). The detection mask is an 8bit grayscale image with the same resolution as the corresponding template. The meaning of region differentiation will be explained further in the text.
Combination of these fundamental features and information about screen element's membership in some defined parental object (panel, menu, screen…) creates a coherent device GUI description. This approach also enables simple functionality expansion. Since display description is a rather specific and device dependant matter, the inspection SW library provides only elementary methods, which are supposed to be used in order to derive more complex functionality for detection and inspection of specific screen objects. Thanks to this quality, the end-user is able to independently boost the provided inspection library with no or minimal support from the inspection system developers.

System set-up and calibration

Individual steps of the system set-up and calibration are illustrated in figures 3 and 4. Overall it is a semi-automated procedure. The required operations are as follows:

- Hardware set-up
  - » Adjustment of mechanical properties: Inspected device positioning; camera/display measuring distance modification.
  - » Adjustment of optical properties and lightening conditions: Mechanical diaphragm set-up and objective focusing, surrounding lightening adjustment.

- Software set-up
  - » Image properties: Shades of gray/color imaging; frames per second rate; gain, white balance and gamma correction adjustment.
  - » Camera calibration: Automatic screen location and projective transformation identification.

Figure 4 depicts windows interface for Screen Inspector SetUp module. In order to simplify the subsequent display inspection the calibration module includes a screen plane to camera frame homography transformation identification [4]. Result of this operation is shown in figure 5 right.

For successful calibration at least 4 points' correspondences between ideal screen content template and image of the screen captured on the camera have to be found.
Consequently the equation (1), which models the mapping between 2D source point [X_S, Y_S, 1] (immediate screen point) and 2D destination point [X_D, Y_D, 1] (point in the camera frame imaging the screen) in homogenous coordinates, can be solved for 8 unknown parameters $h_{11}...h_{32}$ of the projection matrix. Parameter m describes the scale ambiguity of the transformation. Further information on coordinates mapping and the use of homogenous coordinates' can be found for example in [4] and [5].

$$m\begin{bmatrix} X_D \\ Y_D \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix}\begin{bmatrix} X_S \\ Y_S \\ 1 \end{bmatrix} \qquad (1)$$

The calibration points can be defined manually, but the SW environment also enables completely automated approach. The user has only to choose the template image for currently displayed GUI screen. The calibration screen should contain high amount of spatially unique and uniformly distributed interest points. A very suitable example is the 2D pattern of binary noise presented in figure 4, but in most cases a well structured menu screen (as the one from figure 5) should also suffice.

Fig. 4: System setup and calibration GUI, with an example of suitable calibration pattern

The locations of the calibration points in both images and their mutual correspondences are found using "speed-up robust features" (SURF). However, even the SURF method can generate many false correspondences and thus a robust initial estimation of the transformation parameters utilising RANSAC algorithm has been used. Afterwards a simplex method (originally presented in [7]) for function minimization is applied in order to fine-tune the resulting homography matrix by minimizing the sum of absolute differences between the screen template image and the transformed screen image.
The calibration step brings many advantages. Not only that it ensures normalization of the captured

screen images, but it can also significantly reduce inspected area of the camera frames and therefore decrease computational time. Since the calibration is required only at the beginning of the test procedure, its higher computational complexity is not a setback.

Fig. 5: Projection matrix estimation and screen image rectification (left: detected screen area, right: rectified image)

Display inspection

The developed three-level inspection library (see figure 2) implements 3 basic image processing algorithms. These are:

- image similarity check,
- object localization,
- OCR functionality.

More complex tasks can be solved using their combination. This should ensure future extensibility and easy adaptation of the inspection system to different GUI and screen types.

The image similarity check is primarily intended for the comparison between the ideal screen template and the currently displayed content captured by camera. The actual comparison algorithm is based on the so called cosine criterion (for more information see [6]) represented by the equation (2).

$$C_A = \frac{\mathbf{ab}}{|\mathbf{a}||\mathbf{b}|} = \frac{\sum_i a_i b_i}{\sqrt{\sum_i a_i^2}\sqrt{\sum_i b_i^2}} \qquad (2)$$

Compared images are regularly divided into small areas (e.g. 4x4 pixels). Pixel values from these areas are transformed to the one-dimensional vectors **a** (ideal template area) and **b** (displayed screen content area). The resulting criterion value C_A is naturally normalized to the range $\langle -1;1 \rangle$. Because the images have only nonnegative pixel values, the negative results do not occur. Geometrical meaning of the equation (2) is that C_A represents the cosine of an angle between the vectors **a** and **b**. This implies that the criterion is insensitive to linear contrast changes.

An important role during the image comparison has the mask image presented in bottom part of figure 3. Each pixel from this mask serves as a flag register for definition of special meanings of the corresponding screen point. For example setting the most significant bit to 1 is interpreted as a point with no effect to the image comparison computation. This is important if there are continuously changing areas present in the screen. Their actual appearance cannot be predicted and therefore should not be evaluated as a static image.

Other bits from the mask image may have different meanings according to the demands of the tested devices. Their combination creates a greyscale image representation of the mask.

The second basic image processing functionality of the inspection system is the ability to localize the position of separate graphical objects. The algorithm is based on template matching. This method again requires an ideal template possibly complemented by mask image. Localization result of the calculator icon is shown in the figure 6.

Fig. 6: Graphical object (calculator icon) localisation

The ability to read displayed text and numerical data is the third image processing functionality included in our inspection system. A serious shortcoming of today available OCR algorithms is their high sensitivity to font styles. Therefore, the obtained results should be subjected to some kind of post-processing algorithm if possible. A simple example of such post processing is a cross-examination with the expected text content.

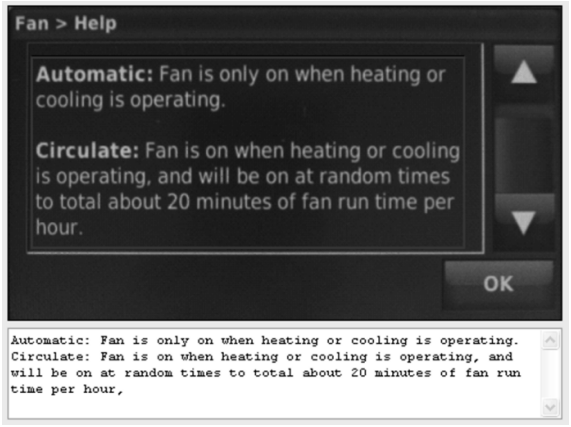

Fig. 7: Utilization of optical character recognition

The OCR algorithm currently utilized in the system is intended for recognition of general texts. It works well with wide variety of different fonts and has the ability to use language dictionaries for corrections. Example of its application shows figure 7.
Despite its flawless operation in the above mentioned example, it completely fails with numbers and characters on fixed segment displays, which are still very common and even appear as a kind of font in many sophisticated devices. This problem has caused the need for adding another OCR method specialized on fixed segment fonts, which is currently under development.

CONSLUSION AND FUTURE WORK

The introduced visual inspection system is intended for effortless semi-automated evaluation of graphical user interfaces. It is based on image processing algorithms, which in combination with user input simulator creates a powerful inspection tool capable of human-like approach to extensive functionality checks of electronic devices.
Due to the designed hardware and software structure of the inspection system, the potential for future expansions and adjustments to new types of tested devices is substantial.
Utilization of computer vision allows complete separation of the inspection system from the tested device and makes it a suitable tool for examination of various screen technologies.
The inspection procedure programming interface is implemented for .NET environment (using CLR C++ and C# languages) and can be distributed as a set of dynamic link libraries. C# and other forms of compatible high-level .NET languages ensure intuitive and rapid development of the inspection programs.
The future development of the core functionality of the system will be aimed at expansion of modularity and specification of standard interfaces for visual data input and output.
Other ways for improvement lie in the development of OCR algorithms and extension of basic image processing functionality.

ACKNOWLEDGEMENT

This work has been supported by the Czech Ministry of Education under the project 1M0567 Centre for Applied Cybernetics.

REFERENCES

[1] S. E. Black, R. L. K.N. Goodman, Wood, Multi-Channel Deep-Memory Digitizing Architecture for Automated Inspection of Large Composite Surfaces, Autotestcon, 2006 IEEE, vol., no., pp.558-564, 18-21 Sept. 2006

[2] J. C. Silva, J. Creissac, J. Saraiva, GUI Inspection from source code analysis, OpenCert 2010, ISSN: 1863-2122, Available from WWW: <http://journal.ub.tu-berlin.de/eceasst/>

[3] G. Bradski, A. Kaehler Learning OpenCV, Sebastopol, O'Reilly Media, Inc. 2008, ISBN: 978-0-596-51613-0

[4] P. Penna, and R. Patterson, Projective geometry and its applications to computer graphics, USA, Prentice-Hall, ISBN 0-13-730649-0

[5] J. Žára, B. Beneš, J. Sochor, P. Felkel, Moderní počítačová grafika. Praha: Computer Press, 1998. ISBN 80-251-0454-0

[6] J. Jan, Medical Image Processing Reconstruction and Restoration: Concepts and Methods. Boca Raton: Francis & Taylor 2006, ISBN 0-8247-5849-8

[7] J. A. Nelder, R. Mead, A Simplex Method for Function Minimization, Computer Journal, vol. 7, pp. 308–313