Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 60 (2013) 243 – 248

Procedia Engineering

www.elsevier.com/locate/procedia

6th Asia-Pacific Congress on Sports Technology (APCST)

Sports monitoring data and video interface using a GUI auto generation Matlab tool

Hugo G. Espinosa[a,*], Daniel A. James[a,b,c], Sean Kelly[a,c] and Andrew Wixted[a,c]

[a]*Centre for Wireless Monitoring and Applications, Griffith University, Brisbane, Qld 4111 Australia*
[b]*Centre of Excellence for Applied Sports Science Research, Queensland Academy of Sport, Brisbane Qld, Australia*
[c]*Queensland Sports Technology Cluster, Brisbane Qld, Australia*

Received 15 April 2013; revised 6 May 2013; accepted 9 June 2013

Abstract

Sport research activities often require graphical user interfaces (GUI) for viewing, merging, tagging and processing of data. More complex processing, like video, requires more complex interfaces but often these GUIs are for short term research use and the investment in creating or purchasing and adapting the GUI is lost. This paper demonstrates an Auto Generation GUI creation tool using Matlab environment. Any GUI is created using a text file that defines all component properties, including the figure, its controls and the callbacks. The auto generation GUI was used in this paper for the creation of a video digitizing interface for monitoring upper arm and forearm rotations of cricket bowlers. This method simplifies the creation and modification of GUIs used in research.

Selection and peer-review under responsibility of the School of Aerospace, Mechanical and Manufacturing Engineering, RMIT University

Keywords: Graphical user interface; video tagging; arm rotation; cricket

1. Introduction

Many research fields require the viewing and manipulation of data using some form of graphical user interface (GUI) as the data is investigated [1,2]. This is not often mentioned in literature since the research outcome is the key result. The GUI will typically combine controls such as buttons, sliders and text boxes with a graphical display used for showing plots, images or video. The output from the GUI can range from the ability to visualize data, to files containing transformed data, extracted data or some form of metadata.

In the Matlab environment, the GUI can be designed using the GUI Development Environment (GUIDE) or using a programmatic GUI construction. Typically this approach starts by creating a figure

* Corresponding author. Tel.: +61-408-709-204; fax: +61-7-3735-5198.
E-mail address: h.espinosa@griffith.edu.au.

Selection and peer-review under responsibility of the School of Aerospace, Mechanical and Manufacturing Engineering, RMIT University
doi:10.1016/j.proeng.2013.07.047

and populating it with components from a graphic layout editor. GUIDE creates an associated code file containing callbacks, which are the functions that execute in response to user generated events, for the GUI and its components. Programmatic GUI files are generally longer, because they explicitly define every property of the figure where GUIDE stores the definitions in its figure rather than in its code file. Although GUIDE offers a 'friendly' development platform, the programmer requires a high knowledge of Matlab to program the callbacks and to use the handles in an efficient way.

This paper presents an alternate approach, an auto generation GUI tool where the graphical interface can be created without touching the GUI Matlab code, one where the user creates a simple text file that defines all component properties, including the figure, its controls and the callbacks. Each component is described by one line of text. The type of object is defined by a keyword followed by the object's name, position and callback function or code to execute. Modifications to the GUI, including additional menu items, buttons and text boxes can be performed through the addition or edition of a single line of text.

In sports monitoring, video or sensor data collected for one study can be reused for other studies requiring development of different GUI tools in an iterative development cycle. This can be complex and time consuming. Using auto-GUI generation tool simplifies the GUI development particularly if it is combined with a specialized data processing toolbox such as the athlete data toolbox (ADAT) [3]. The auto generation GUI together with the ADAT processing toolbox will allow non-technical researchers to monitor biomechanical information through a user-friendly platform.

Prior to this paper, the auto generation GUI has been used for the creation of a number of graphical based tools including tools for synchronizing data from multiple sensors, motion capture and video [4]. The example described in this paper is a video digitizing interface for monitoring upper arm and forearm rotations of cricket bowlers, by using a marker-based system that pointed at the bowler's wrist and elbow from every high speed motion video frame [5].

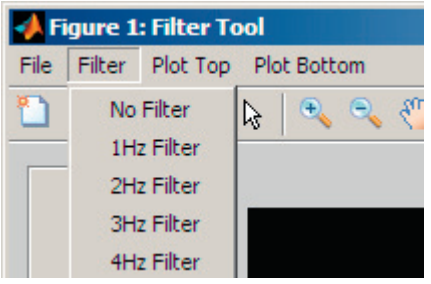
2. GUI auto generation tool

The GUI Auto Generation tool was developed to simplify the development of GUIs in Matlab. This tool reads a text file and generates the GUI based on the controls listed in the text file. Each line of the text file represents one GUI control, such as a button, slider, axis or menu item. The line also includes the name of the function to use when a control is selected. A number of controls are supported: Multi-level menus, multiple axes in a GUI, standard menu and toolbar, buttons, sliders, panels, and editbox.

Each control is described by a keyword such as "ADDBUTTON". The GUI auto generation tool parses the input text file and finds all the lines starting with the specified keywords. Where the ordering of the keywords is important, as in the case of the hierarchical menus on the menubar, the order of definition is maintained. Once the text definition file is parsed to extract the required controls, the details for each control are extracted. With the parsing complete the auto generation GUI creates the layout of the GUI from the definition in the text file and defines the code or function that will run on a button press. Once a few GUIs have been produced using the tool, a number of definition templates exist that speed up the generation of subsequent GUIs. When run, the GUI Auto Generation tool creates a script that defines the menu structure, another script that defines the controls and a third script that manages the layout of axes within the GUI. If the program requires multiple figures each having the same menus or buttons or layout, the code that opens subsequent figures can simply call the pre-created scripts. All figures can then share common layouts where desired.

All controls are defined by a keyword, some of these keywords, such as STDTOOLBAR and STDMENUBAR do not require any further definition. Menus require the MENU keyword, the text to display and the code to execute or function to call when the menu is selected. Most other controls will also require position and size information. In the case of menus, the keyword "MENU1" is used to define a top-level menu with "MENU2" used to create submenu items associate with the preceding top-level

menu item. A MENU3 keyword also exists allowing the definition of submenus belonging to a level 2 menu. The order the menu items are defined in the text file determines the order they appear in the menu system. As an example, Figure 1a shows a snapshot of an Auto Generation Matlab figure menu. Figure 1b shows an extract of the text editor lines used to generate the menu. In the example the MENU1 instruction is used to generate the top level menu items "File, Filter, Plot Top and Plot Bottom", the MENU2 instruction generated the items for the "Filter" submenu, in this case "No Filter, 1Hz Filter, 2Hz Filter and 3Hz Filter".



(a)

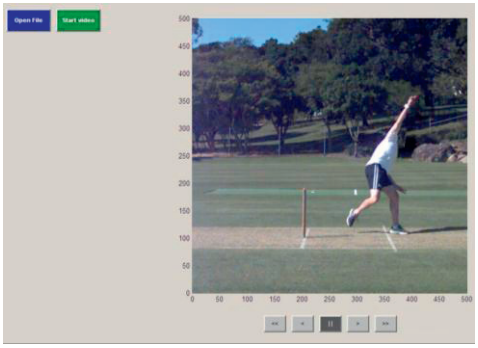
```
STDTOOLBAR
MENU1;File;
MENU1;Filter;
MENU2;No Filter;
MENU2;1Hz Filter;
MENU2;2Hz Filter;
MENU2;3Hz Filter;
MENU1;Plot Top;
```
(b)

Fig. 1. (a) Snapshot of an Auto Generated Menu. (b) Extract of text file that created the menu. For clarity the function callback has not been shown.

Typically a GUI will include plots and buttons and various controls place in strategic locations. In the creation of a video tool, the first step is to create an empty figure plot followed by adding the control buttons used to manipulate the video. Every button can be added by a program line using the following format:ADDBUTTON<*button number*>;left;bottom;width;height;<*button text*>;<*code to execute*>; Where "left;bottom;width;height" refer to the coordinates on the figure and dimensions of the button. For each button added it would be usual to write a button callback function. The name of the callback function is given on the button definition line. In the example given in Figure 2 the first button "Open File" has the callback function OpenFile( ) which is passed the variable GUI (which contains the handles to all graphics objects – in case they are required for manipulation). The order the buttons appear has no relation to their position. The button number corresponds to a handle number. Figure 2a shows a snapshot of the creation process of a video digitising GUI and Figure 2b shows the GUI definition file with the line instructions that create the buttons. It is not strictly necessary to associate a callback function with the controls, any valid Matlab code that does not include the ";" character can be executed. A "Grid On" button might just execute the code 'grid on'.

If a GUI has been developed using this method and it is determined that an additional control is required then it is simple three step process to add the new control. (a) Decide the details of the control such as where it is to be located. (b) Add the control definition line in the GUI definition file, and (c) write the callback function. When the GUI auto generation tool is re-run, the new button exists and should be functioning.

Depending on the program complexity some program architecture decisions may need to be made. The GUI under construction in Figure 2 used a global variable VIDEO, which was a structure holding data relevant to the display of the video images. There was also a variable associated with the GUI auto generation tool called "GUI". This variable contained the handles to graphical objects created by the auto generation tool. In the example code of Figure 2b, the GUI object was passed to functions so handles to objects could be made available in any function. These are just programming options demonstrating the choices available. Some GUI sport analysis tools did not require any passing of handles and the code executed on selecting a button or menu item used a mix of Matlab code, scripts or functions.



(a)

```
ADDPLOT1; 175; 65; 335; 335;
%Button;        Position;    Label;    Callback
ADDBUTTON1;10;385;60;20;'Open File';openFile(GUI);
ADDBUTTON2; 80; 385; 60; 20;'Start Video';
startVid(GUI);
ADDBUTTON3;260;10;30;20;'<<';stepBack2(GUI);
ADDBUTTON4;300;10;30;20;'<';stepBack(GUI);
ADDBUTTON5;340;10;30;20;'||';btnPause(GUI);
ADDBUTTON6;380;10;30;20;'>';stepForward(GUI);
ADDBUTTON7;420;10;30;20;'>>';stepForward2(GUI);
```

(b)

Fig. 2. (a) Snapshot of the video interface creation process. (b) Extract of text file that created the figure with the control buttons.

3. Application: Monitoring arm rotation of cricket bowlers

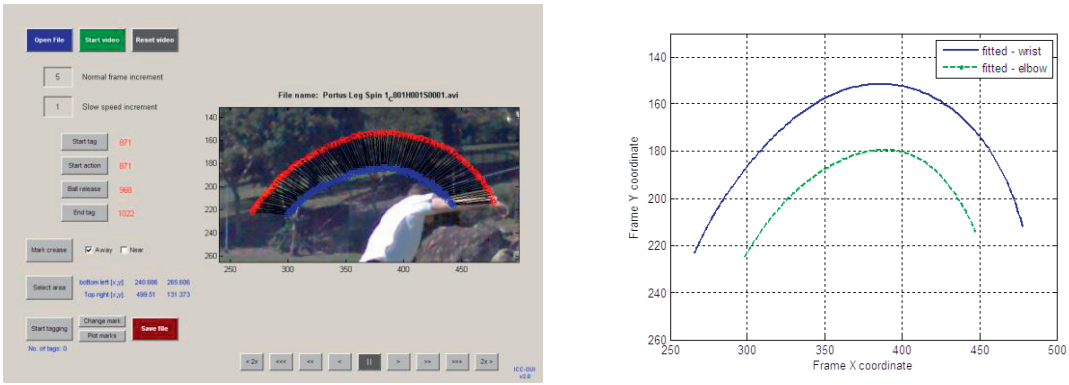
Athletes are usually monitored with inertial sensors (accelerometers and gyroscopes), high speed video and motion capture. The advent of inertial sensor monitoring created a new monitoring paradigm and brought with it the necessity to interpret these sensors in the context of existing technologies. It is useful to merge these multiform data to extract the parameters of elite technique and develop an understanding of how inertial and traditional measures are related [4].

The upper arm and forearm rotations are critical in a number of sports. Monitoring these rotations is essential for the enhancement of the athlete's performance, since it contributes to the speed of the serve in tennis, to the propulsion of swimmers and to the bowling action in cricket, among others. A GUI was developed for this application to allow the digitising and merging of video and sensor data for the cricket bowling arm action [5].

The video digitising interface was developed using the GUI Auto Generation Tool and is shown in Figure 3. The interface allows the user to enter manually the number of frame increments in normal and slow speed, the start and stop tagging frames and other features such as the ball release frame, crease coordinates and stump location.

The monitoring of the upper arm and forearm rotations of cricket bowlers was performed by tagging markers on the bowler's wrist and elbow from every high speed motion video frame (Figure 4a). Using transformations, the positional information can be converted to acceleration and pattern matched with the accelerometer signal. Figure 4b shows a high-order polynomial interpolation of the elbow/wrist data obtained from the marker-based system.

Fig. 3. Video digitising interface snapshot of the elbow‘s/wrist‘s marker-based system.



(a) (b)

Fig. 4. (a) Video digitising interface snapshot of the elbow‘s/wrist‘s marker-based system (b) High-order polynomial interpolation of the elbow/wrist data.

An associated software using the GUI Auto Generation Tool was developed to automatically detect *overs* and *deliveries* from 4 inertial sensors (2 upper-arm and 2 forearm) with 3 channels each, consisting of a 3D accelerometer and 3D rate-gyroscope on the cricket bowler’s upper arm and the same on the forearm, where a delivery can be selected and merged with the video data. Figure 5a shows a 3D accelerometer and 3D rate-gyroscope raw data. ADAT toolbox [3] was used to perform the signal processing (Hamming filtering and combined vectors) necessary for the bowling detection to function (Figure 5b). Figures 5c and 5d show the GUI interface for the automatic detection of overs and deliveries, respectively.

The auto generated GUI allowed the quick matching of video and sensor inputs for bowling deliveries. Sensor and video data was able to be synchronised and redisplayed allowing interpretation based on the motion causing the signal. Fast creation of GUIs speeds the research task.

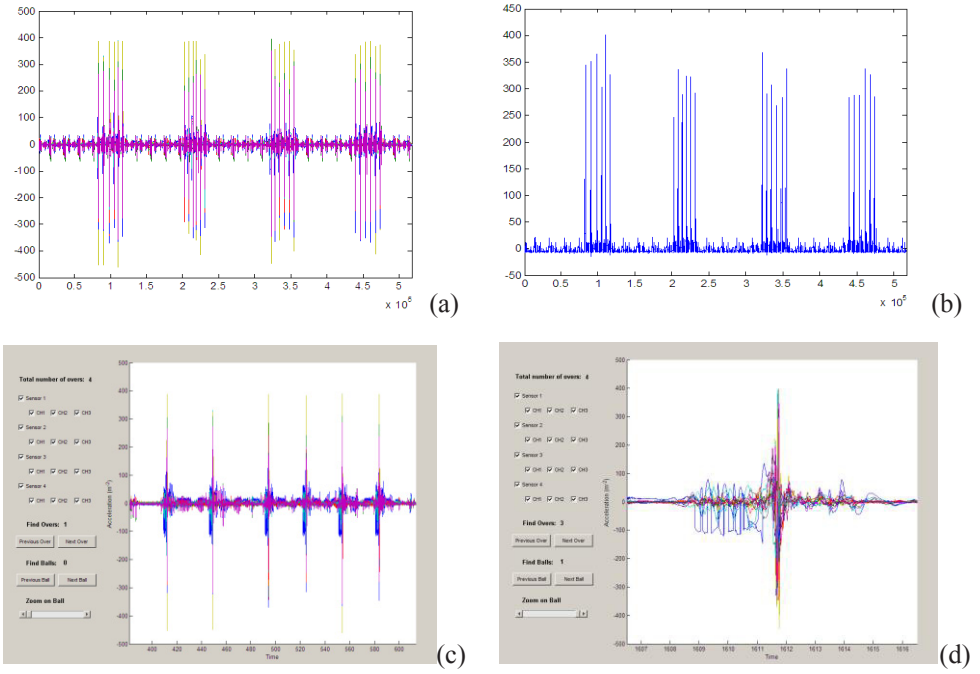


Fig. 5. (a) 3D accelerometer and 3D rate-gyroscope raw data (b) Hamming filtering and combined vectors (c) Graphical user interface for the automatic detection of overs and deliveries from 4 sensors with 3 channels each (d) Selecting and zooming on one random delivery from the 12 channels.

4. Conclusions

Development of GUI tools for sport and other forms of research is costly in time and funds, and GUIs developed for one research task may be discarded at the completion of the task. This paper has described a simple process developed to aid GUI creation. GUIs can be quickly, defined, created, used and discarded. The creation of one GUI creates templates than can be easily modified to create another GUI.

As an application, the auto generation GUI was used for the creation of a video digitizing interface for monitoring upper arm and forearm rotations of cricket bowlers, together with an interface for the automatic detection of *overs* and *deliveries* from inertial sensors.

References

[1] Xinguo Y, Farin D, Current and Emerging Topics in Sports Video Processing. *ICME IEEE International Conference on Multimedia and Expo*, 2005; 526-529.

[2] Rowlands DD, McCarthy M, James DA. Using inertial sensors to index into video. *Procedia Engineering*, 2012; **34:598**-603.

[3] James DA, Wixted A. ADAT: A Matlab toolbox for handling time series athlete performance data. *Procedia Engineering*, 2011; **13:451**-456.

[4] Wixted A, Portus M, Spratford W, James, DA. Detection of throwing in cricket using wearable sensors. *Sports Technology*, 2011; **4:134**-140

[5] Espinosa HG, James DA, Wixted A. Video Digitising Interface for Monitoring Upper Arm and Forearm Rotation of Cricket Bowlers. *Proceedings of ASTN1,* 2013; **1:24**-26.