Interface on Interaction with 3D Medical Data

Lukáš Tencer*
Faculty of Mathematics Physics and Informatics
Commenius University
Bratislava / Slovakia

Abstract

The 3D data, processed in radiology, requires interaction metaphors different from common user's needs. In cooperation with medical institutions we have designed a Graphic User Interface (GUI) and Human Input Device (HID), that both address the special requirements of the users. For the GUI we designed new elements to enhances customizability. As a HID we developed an alternative approach for the commonly used mouse, employing a web camera.

Keywords: Graphical User Interface, Human Input Device, Motion capture

1 Introduction

In the beginning of our work we did research in medical institutions. We were consulting radiologists and laboratorians to gather their requirements. They are working with 2D slices and 3D reconstructions. Most of the time they need to inspect and qualify data using exact tools as sedimentation measurement, distance measurement and others [6]. Our work is aimed at 3D reconstructions, so for our purposes we will use classic 3D models represented as meshes as testing data.

Resulting from the discussion we came up with two main requirements: faster access to tools and better orientation in views. To achieve faster access to tools we designed the myMenu and myCollection widgets and to enhance orientation in the views the Viewbar was designed.

We also created a concept for an alternative input device. A web cam and painted glove are used to simulate mouse movement and events. The concept was assessed as good by the target group.

The paper is organized as follows. In Section 2, we describe existing solutions and standard concepts we use and closely specify functionality and design of the implemented widgets. In Section 3, we present our GUI concept, as well as some improvements of commonly used concepts. In Section 4 our HID solution is introduced. In Section 5, we show the obtained results and state the opinions on software gathered from the target group. Finally, in Section 6 we present ideas suitable for further expansions of existing work.

2 Background

Presently, many solutions like Osirix [10] are used for similar purposes, mainly delivered with special hardware. We discuss existing solutions and try to improve them according to the requirements from the users. We present three new major widgets and some improvements of existing ones.

Existing solutions use several monitors to extend the workspace, and the ideal number of monitors according to the opinion of some developers is 3. One is used for the agenda second one for slices and the third one for reconstructions. But usually two monitors are used in practice with agenda and slices being handled on one monitor. We propose a solution with only one monitor, because our program is focused on 3D reconstructions of the data.

We designed the GUI according to the guidelines which can be found in [2] or [4]. In [11] a HID similar to our solution is presented, which is used for face tracking. However, the authors do not propose the use of a higher level library.

For our purposes we use OpenGL as we do not care about rendering or visibility. Also we use OpenGL for handling window events and controlling inputs from keyboard and mouse. DirectX or Open Inventor, could be also used, but OpenGL fits for our purposes best.

The standard concept we build on is WIMP [7] (Windows, Icons, Menus, Pointing device), which is the dominating GUI form as of today. WIMP concept means that a program is placed in a standard window with events as resizing or moving in a workspace. Access to tools and dialogs is given by icons and/or menus. Everything can be controlled using a pointing device, optionally in combination with other devices. Widgets and their functionality and design, are described in more detail in Section 3.

For the HID, we will use OpenCV [12], a higher level library, which implements many algorithms for computer vision and image processing. Many solutions similar to ours exist [11], but they mostly concentrate on moving of the mouse pointer not on handling the events.

We use the algorithms from OpenCV library to identify a hand in a picture and to track the motion of the hand in a sequence of frames. CamShift, the widely used algorithm, is suitable also for our purposes, the only problem is with fast hand motions. We can remove this

* lukas.tencer@gmail.com

problem using repeatable search after a loss of the tracked component in the picture.
The problem can be divided into two parts. First we need to recognize and track hand motions and then we process events and recognize the gesture. The whole concept is closer described in Section 4.

3 The GUI concept

When designing the GUI we have to keep in mind the two main requirements, which were obtained from the target group. The first one is to improve the accessibility of the used tools and the second one to improve orientation in views on 3D object.
We use a standard window concept as we know it from common operating systems [5]. The layout of widgets is typical, with the exception of a side panel on the left side, as you can see on Figure 1. It was placed here in order to be quickly accessible everywhere from the workspace.

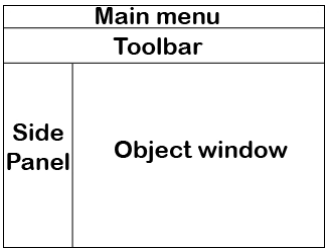

Figure 1: Window layout

The whole GUI is based on a layer system. A range of levels X = (0…n) is given to the program. Every widget is associated with subset of these levels $X_i = (x_i…y_i)$ and also a concrete level x = c is given to each part of a widget. When the GUI is drawn, first the widgets and their parts are sorted depending on the given level and after that they are displayed from the lowest to the highest level. This principle is illustrated in Figure 2. Also the GUI is designed to be fully externally skinable, which means that for changing the look no interaction with the code is needed.

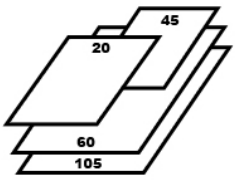

Figure 2: Layer principle

Two common widgets are used: the Main Menu on the top side and the Toolbar placed under the Main Menu as proposed in [3]. The only change done is on the Toolbar (see Figure 3). It is designed to have more layers to include more tools. Between layers you can navigate using the arrows in the center or you can view all levels by clicking on the arrow at bottom right of toolbar. You can use a tool by left clicking on it in the Main Menu or the Toolbar. By right clicking on tool you can add it to a specialized widget.

Figure 3: Toolbar

First widget is myMenu, which is connected to the Main Menu. Items could be added to myMenu by right clicking on them in the Main Menu, you can also remove items from myMenu by right clicking on them in myMenu. Items can be sorted alphabetically or in order of addition. The whole myMenu is shown after clicking on myMenu button in the Main Menu, as you can see in Figure 4. It also replaces a classic context menu, that means it is shown after a right click somewhere in workspace. It is used to gain faster access to frequently used tools.

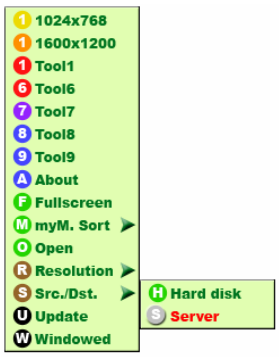

Figure 4: myMenu

The second widget is called myCollection. Its functionality is the same as in the case of myMenu, but here the Toolbar replaces the Main Menu. There is one major difference; the myCollection is called by the middle mouse click. Also look is changed, for myCollection a circle menu is used (see Figure 5). Basic purpose of myCollection is to get easier access to the tools from the Toolbar.

Figure 5: myCollection

Another requirement was to design a tool to improve orientation in views and allow quick change between views. For this we use the panel on the left side of the window called Viewbar (see Figure 6). Here an actual state of scene including camera, object or clipping plane position can be placed. By a simple click on the button in the top left corner a view is added to the Viewbar and it could be restored by a left click on it or removed by a

right click. It is an easy way to store interesting and important views. The viewbar can also be hidden to enlarge workspace.

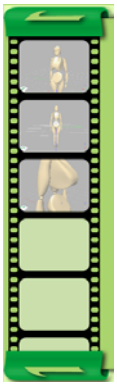

Figure 6: Viewbar

We have experimented with the orientation in 3D and designed some concepts. The first one is an orientation tool in the bottom left part of the window, where the position of camera (eye) is shown relatively to the object we are looking at (see Figure 7). The second one is handling objects, when the transformation is first done on a pivot (cube) representing the object, and after that the object itself is transformed (See Figure 8a). The last one is placing a clipping plane, which is implemented as moving plane on a surface of a ball (see Figure 8b). Using this concept we can achieve the placement of the clipping plane in any place and any direction.

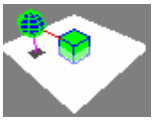

Figure 7: Orientation widget

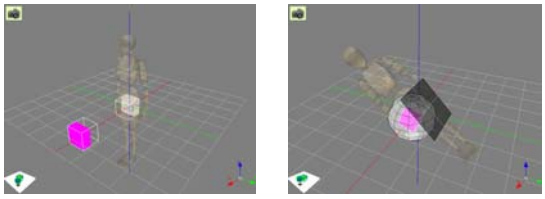

Figure 8: a) Moving object b) Placing clipping plane

4 The HID concept

We designed a device to control mouse movement, as well as events (right click, left click). We use two main components: a painted glove and a web camera. We use gestures to control events and track the glove to control movement.

The glove is painted with two colours. We tested nine bright colours under different light conditions (artificial light, natural light) and in different places (interiors, exteriors). We have achieved the best recognition results with bright pink and bright green. Pink was selected for the larger region on the hand and green for the thumb region, which will serve for recognizing the mouse clicks. Gestures for left and right mouse click were implemented. A left click is executed by the moving thumb up and down while other fingers are closed to fist and the hand is in a vertical position. The gesture for right click is similar, differing only by the rotation of the hand, which is in a horizontal position here.

We need to fulfil three tasks to successfully control the mouse using web camera. These tasks are: finding the hand in the picture, tracking the hand in the picture and recognizing the gestures. To simplify these tasks we are using the OpenCV library, which implements algorithms suitable for our needs. The resolution of the pictures we are working with is 320*240. This value is sufficient for the movement tracking to be accurate, yet small enough to allow processing almost in real time. On AMD Turion 64 X2 TL-60 2.01 GHz response time is the same as when we use direct input device as USB mouse.

The processes for the hand and thumb region are identical, only differing in colour range. Thus we exemplarily illustrate the process for the hand region.

The process mainly consists of two parts. First the glove has to be identified in the input stream. In the second phase the glove is being tracked. The input sequence consists of frames in RGB (Red, Green, Blue) colour space, which is converted to HSV (Hue, Saturation, Value) [8], where the range of a particular colour can easily be identified and selected. In the converted stream pixels in the range for pink are selected and used to create a binary mask (see Figure 10). The mask is eroded twice and dilated once to get rid of small areas and noise. After that the largest contour in the picture is searched, which is supposed to be the glove. Now that the largest contour is marked (see Figure 11), we know that the hand is visible and can be tracked.

Figure 9: Input image

Figure 10: Ranged image

Figure 11: Binary mask of largest contour

For tracking an object in series of frames we use the CamShift object tracking algorithm [1] from the OpenCV library. First, it finds the center of the object using the MeanShift [13] algorithm and after that it calculates the area and orientation of the object. As a result we can get the number of iterations made within MeanShift.
CamShift is a modification of the MeanShift algorithm, which is a robust method for finding probability distributions. It tracks the center and size of a probability distribution of an object. Its accuracy is based on probability distribution of an object, which we get by using the histogram and calculating the backprojection of an image. We calculate the histogram only for the part of the image under the binary mask from the previous step (see Figure 12), after that we calculate the backprojection [9] using our histogram (See Fig. 13).

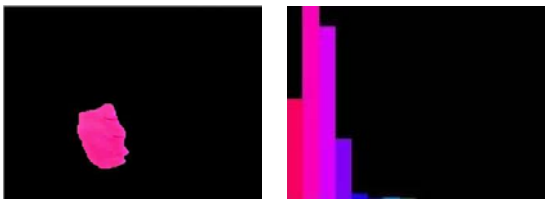

Figure 12: Largest contour and its histogram

Figure 13: Backprojection

As a result of the tracking process in every step we get object aligned bounding boxes for both, hand and finger (see Figure 14 with inscribed ellipse to bounding box). Based on this information we move the mouse pointer and execute events. The mouse position is derived only from the center of the hand. It needs to be modified, so that the thumb is visible also when the hand is in top position.

Figure 14: Hand found in image

Due to fact that we are using low resolution images for detecting the object on the one hand, but a higher resolution screen on the other hand a small change in the detected coordinates caused by hand shaking or CamShift abnormality, can result into large change of absolute mouse pointer coordinates. To handle this we calculate the coordinates as a moving average of n successive values. We also have tested other types of averages, but they were slower or less accurate as the moving average. We calculate the average value of $x_i...x_n$ values and when a new value x_{n+1} is measured we remove x_i and add x_{n+1}, so new sequence is $x_{i+1}...x_{n+1}$. The average value for time step i is calculated as:

$$x_i = \sum_{j=i-n+1}^{i} \frac{x_j}{n}$$

Gestures are recognized using both, hand and thumb bounding box center and size. We are detecting four events, which are mouse up and down for each mouse button. First of all we compute polar coordinates of the thumb center depending on the hand center. We detect events only when the thumb center polar coordinates are approximately 0 or $\pi/2$, to avoid clicking while changing the orientation of the hand. Mouse up/down events are recognized using the change of the coordinates of the hand and thumb centers. When the ratio between the change of the absolute value of the x-coordinate of the hand respectively thumb center is greater than a threshold a left mouse button event is triggered, where up or down is based on the current mouse button state. The detection is similar for the right mouse button, but we use the y-coordinate instead. The threshold value is obtained from the size of the bounding box, scaled to correspond with the real events. These are all gestures which are recognized for now.

5 Results

Both GUI and HID solutions were presented to the target group. Both concepts were positively accepted. After a discussion we came up with conclusion, that the widgets used in the GUI can speed up the work and improve the accessibility of the tools. Concretely myMenu and myCollection were most discussed and requested to be

implemented in future versions of actual versions of used programs. Because there is no exact procedure to measure the effectively of these new tools we can only wait for user responses, while they will use it in praxis. In the case of the HID reactions were also positive, but here much more work needs to be done. One major achievement was development of exact process to identify color regions in the picture and track them. Other success was effective and fast method to stabilize mouse pointer while it is controlled by glove, using moving average. Next chapter brings some suggestions arisen from the discussion with the target group.

6 Future work

The technologies used for the GUI are satisfactory enough, widgets as concepts are usable in any larger context. In the GUI we also experimented with the look of elements, but in some cases these experiments were not successful, as in the case of highlighting for myMenu. As we have mentioned, the GUI is skinable so the look can easily be adjusted to fit different needs.

The technologies used for the HID could be improved. Especially some modifications of the detection algorithm might be useful, because after fast hand motions the tracked object can be lost. As for now we solve this problem by doing repeatable search for the object just as the initial search phase.

There are several ways for the HID to be extended. First, every finger can be painted with a different colour, but here a pattern recognition technique for recognizing gestures will be needed. Second, both hands can be used for orientation, which will help to implement a 3D input. Next, we can use a "guitar metaphor", where one hand is reserved for a keyboard and the other hand serves as a device, from which we can accept events as from a joystick. There could be 5 events detected - for each finger one event and the events could have different meanings based on position and orientation of the device. In this case a positioning device could be used instead of a web-cam.

References

[1] **Gary Rost Bradski,** *Computer vision face tracking as a component of a perceptual user interface. In Workshop on Applications of Computer Vision*, pages 214–219, Princeton, NJ, Oct. 1998.

[2] **Jeff Johnson**, *GUI Bloopers 2.0: Common User Interface Design Don'ts and Dos*. Morgan Kaufman, Sept. 14. 2007

[3] **Kiger, John L**., *The depth-breadth trade-off in the design of menu-driven user interfaces. International Journal of Man-Machine Studies*, Vol. 20, 1984, pages 201-213.

[4] **Norman, Kent L**., *The psychology of menu selection: designing cognitive control of the human/computer interface.* Norwood, NJ: Ablex, 1991.

[5] **Cooper, Alan, Robert M. Reimann, and David Cronin,** *About face3: the essentials of interaction design*. New York: Wiley, 2007.

[6] **Jimman Kim, David D. Feng, Tom W. Chai,** *A web based medical image data processing and management system. ACM international Conference Preceeding Series; Vol. 9*, pages 89-91, Sydney, Australia, 2000.

[7] **Ashley George Taylor,** *WIMP Interfaces (winter 1997),* [online: 30.1.2009], http://www.cc.gatech.edu/classes/cs6751_97_winter/Topics/dialog-wimp/

[8] **Raphael Gonzalez, Richard E. Woods,** *Digital Image Processing, 2nd ed.*, Prentice Hall Press, 2002

[9] **Frank Natterer,** *The Mathematics of Computerized Tomograph,* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001

[10] **The Osirix Foundation,** *OsiriX Imaging Software,* [online: 30.1.2009], http://www.osirix-viewer.com/

[11] **Alejandro Rivero,** *Mouse Webcam,* [online: 30.1.2009] http://dftuz.unizar.es/~rivero/alumnos/vmouse.html

[12] **Gary Rost Bradski, Adrian Kaehler ,** *Learning OpenCV: Computer Vision with the OpenCV Library*, O'Reilly Press, Sep. 2008.

[13] **Dorin Comaniciu, Peter Meer ,** *Mean Shift: A robust Approach Towards Feature Space Analysis*, IEEE Transactions on Pattern analysis and Machine inteligence, Vol. 24, May 2002