A Marker-free Visual Based Interfacing Device for Menu Driven Systems

Willem Visser *Yuko Roodt* *Willem A. Clarke, Member, IEEE*

University of Johannesburg
glasoog@gmail.com

Highquest, Johannesburg
yuko@highquest.co.za

University of Johannesburg
willemc@uj.ac.za

Abstract

In this paper we propose a marker-free visual based interface device to be used with menu driven systems. This system, called HandsFree, uses the Graphics Processing Unit (GPU) together with Shader technology to perform the image processing. HandsFree makes use of a web camera to gain user input and can be used in an array of surroundings. This paper focuses on the architectural design and testing of the system.

1. Introduction

Visual Based Interfacing (VBI), also known as Vision Based Interaction, has gained interest in the past couple of years as it does not require users to have any hardware (except for a camera or a sensor) to give input to the computer. The visually obtained input is in the form of human movement. The two ways of tracking movement are by using body markers and tracking without markers [1]. Using markers for tracking simplifies the job, but is invasive and time-consuming as the user has to apply the markers each time before using the system. Marker-free tracking on the other hand gives the user total freedom of movement. Marker-free tracking is usually model-based and input can be retrieved from one or sometimes multiple cameras.

The EyeToy [2] that was specifically developed for the Sony Playstation 2 is an example of a VBI device that does not make use of markers. The commercially successful EyeToy is a colour USB camera which is placed on top of or directly below the television so that the user can be seen on the television screen. The EyeToy requires users to use their body as the input device. The elements of the game are augmented over the recorded view. In the EyeToy: Lemmings game, as seen in Fig. 1, the user moves his arm to form a bridge for the lemmings to walk over.

Figure 1: *Screenshot of EyeToy: Lemmings being played.*

The next section will follow with an overview of how other researchers have visually obtained input without the use of markers. It will be followed by an overview of the marker-free VBI device, HandsFree, which is proposed in this paper. This section will also look at how HandsFree works. A short explanation of the setup of the web camera will then follow. In section 5 the architectural design will be given and all the elements of the system that has to do with the processing of images will be explained. Following this the experimental setup will be discussed and this will be followed by the experimental results as well as a discussion of the results. The paper will finish with concluding remarks.

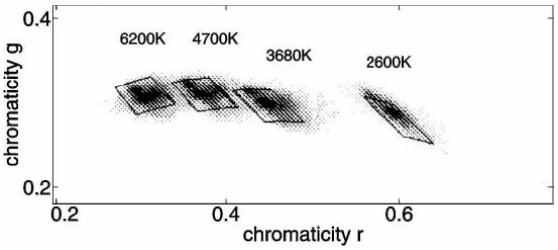
2. Related Work

Research done by different people each found different ways of visually obtaining user requests. Hansen et al. [3] focused their research on tracking the human eye. They provided an improved likelihood model to cope with major local and global lighting changes and made use of an infrared camera to obtain their input. Manresa-Yee et al. [4] uses a standard web camera to obtain their input, which is in the form of eye winking detection and nose tracking. By tracking the nose they developed a system that replaces the use of the mouse. What makes it different from the Camera Mouse is that it uses the detection of eye winks to replace mouse button clicks.

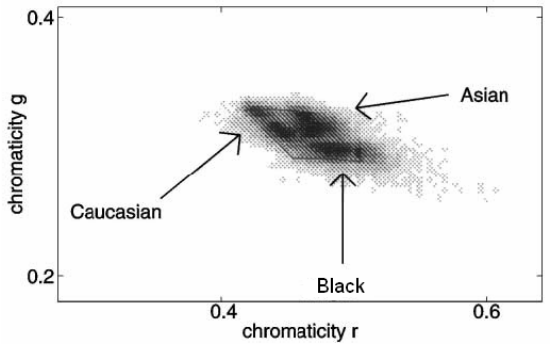
Another popular feature to track is the human hand, as it is a natural interface device for human beings. Kölsch et al. [5] presented a fast hand tracking method that is robust against indoor and outdoor lighting and dynamic backgrounds and that can be used by different people. Their flock of features method uses KLT (Kanade, Lucas and Tomasi) features, which detects a feature as an area with a steep brightness gradient along at least two directions. They use this together with foreground-background separation where the hand is the foreground. The separation is done by looking at the normalised RGB histogram of the hand area together with a horseshoe-shaped area around the hand and by doing skin colour detection of these two areas. The skin colour of the specific user is learned as the user uses the system and is not done *a priori*. The learning algorithm is a variation of that developed by Störring et al. [6].

Skin colour detection is very important in the field of VBI. Research presented by Störring et al. in 2001 offered a robust skin detection algorithm which provided a huge breakthrough in human-computer interaction. They discovered that if an r-g plot is made of all the pixels in the image, where r = R/(R+G+B) and g = G/(R+G+B), then the skin coloured pixels are grouped

together in a Skin Area in the plot. They also discovered that the Skin Area is dependent on the Correlated Colour Temperature (CCT) of the light source or combination of light sources. The Skin Area is slightly different for different people and particularly for different races of people, as can be expected. Fig. 2 illustrates these phenomena.



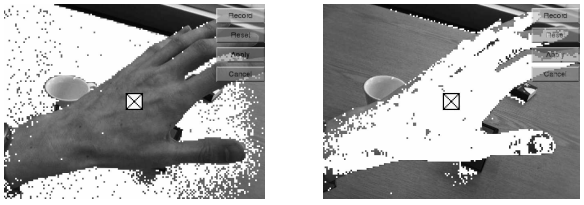
(a)



(b)

Figure 2: *(a) r-g Plot of Caucasian skin under four different illuminations (the solid line shows the skin colour area, modelled for each candidate); (b) r-g plot of three different races' skin under an illumination with CCT = 3680 K.*

Although the Skin Area is different for different candidates and under different light sources, an adaptation algorithm can be used together with the above mentioned theory so that it can be used in a VBI application, as was done by Gunther Heidemann et al. [7] in their research for the VAMPIRE project. By recording only four or five images of the user's hand, they can model the skin colour successfully. To show the pixels that fall within the Skin Area, they change the pixel's colour to white. They used a Head Mount Display (HMD) as their output device and two stills from the HMD of the skin detection are shown in Fig. 3.



(a) (b)

Figure 3: *(a) Untrained skin detector detects wood as skin; (b) Trained skin detector.*

Detecting skin colour is very useful when only the hand or face needs to be tracked, but it is sometimes needed to track the whole body, as in the case of human motion tracking.

The easiest way to do marker-free human motion tracking is to use a stationary camera and to extract the moving human silhouette from a background image. The background is removed using background subtraction. Devlaeminck [8] in his masters dissertation on human motion tracking presented a system that is based on Zimmermann and Svoboda's probabilistic estimation of human motion parameters. The system made use of 12 cameras in a room. These cameras were positioned in such a way that the user can be recorded by at least three of the cameras at any given time. The input from the cameras was then used to position a model of the user in a 3D virtual space in such a way that the model mimics the user.

Han et al. [9] used both colour and infrared cameras to better the technique. By incorporating an infrared camera, which operates in the long wave band of 8-12 µm, they record a thermal image with pixel values representing temperatures. The thermal camera has the advantage that lighting conditions as well as the colour of the human's clothes and skin are disregarded. Also, the temperatures of the human are usually significantly different from that of the background, making this a robust type of detection.

3. An Overview of HandsFree

The problem with skin colour detection is that the system has to be retrained for each new user. The VBI system proposed in this paper will make use of learning the background as the system can then be used by any user. This is better than the method of background subtraction as HandsFree allows for slight change in the background, even while the user operates the system. HandsFree will specifically be used with menu driven programs. The system consists of a web camera, a projector or computer screen and a computational unit. The web camera is a less expensive solution than an infrared camera, with good results still being obtained through the use of localized averaging. A model of the system is shown in Fig. 4.

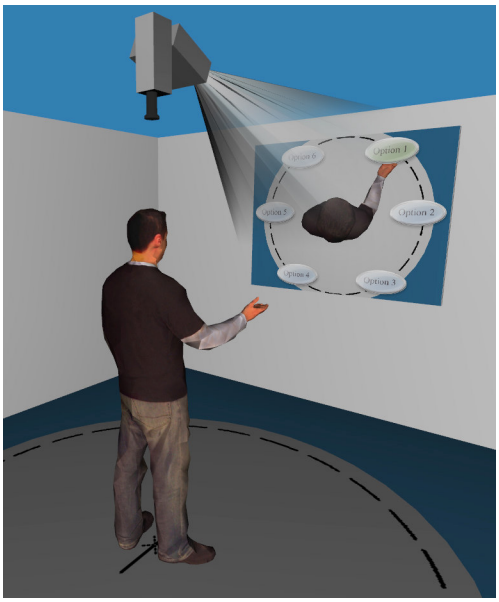


Figure 4: *Design of HandsFree.*

To use the system, the user must walk underneath the camera. The user is then recorded from above using a video camera that is mounted against the roof. The live video feed is then projected together with overlaid menu options against a white wall or on screen. What the user views on the screen can be compared to the face of a grandfather clock. Users are the middle of the clock with their arms being the arms of the clock. The menu options are displayed around the user like the numbers of the clock. But instead of there being twelve options, like that of a clock, there can be any number of options being displayed, up to a certain limit. Users can then choose a menu option by pointing one of their arms in the direction of the desired option. The option will then be chosen and the next level of options in the hierarchy will be displayed. HandsFree also has the capability of calling a specified DLL or web service function when a button is pressed. The names these functions, their locations and parameters, as well as the menu hierarchy are all specified in the System Design XML file. Button types are used to distinguish between different types of buttons that can be used in HandsFree. An example file is shown in Fig 5.

```
<?xml version="1.0"?>
<button text="RootNode">
   <button textboxVisible = "0">
   Start Application
      <button btnType="0" textboxDefaultVal="Calculator" textboxRetrieveLayer = "3">
      Calculator
         <button dllName = "SimpleCalculator" funcName = "addNumbers">
         Add
         </button>
         <button dllName = "SimpleCalculator" funcName = "subtractNumbers">
         Subtract
         </button>
         <button dllName = "SimpleCalculator" funcName = "multiplyNumbers">
         Multiply
         </button>
         <button dllName = "SimpleCalculator" funcName = "divideNumbers">
         Divide
         </button>
         <button btnType="4">
         Keypad
         </button>
         <button btnType="2">
         Main Menu
         </button>
         <button btnType="3">
         Exit
         </button>
      </button>
   </button>
</button>
```

Figure 5: *System Design XML file*

HandsFree works as follows: It learns the background and detects sudden changes in the area where buttons are located. To do this the background is divided into 64x64 blocks and each block is averaged. The averaging technique was introduced as there are small differences from one frame to the next due to small vibrations and due to the image quality from the web camera. "Significant change" is when the colour at a pixel is outside a pre-defined range around the average. The button-area that shows the most change and is above a certain threshold is marked as selected. If that button keeps the status "selected" for more than a pre-defined time, the button is pressed. As users are also gradually learnt into the background, their previous choices will be remembered and this will interfere with the future use of the system. To correct this, the learnt background is reset whenever a new set of buttons is displayed. The usability and restrictions of HandsFree depends on its pre-defined colour range and selection time and the ideal settings for these parameters will be investigated in sections 6 and 7.

4. Setting Up the Web Camera

In order for the system to work properly with a specific user or group of users, it is important that the setup of the system is done correctly. There are a lot of factors that come into play when setting up the system. In Fig. 6 the setup is shown together with the factors that need to be set for the system to work correctly.

In order for the system to work correctly it is important that the user is in the middle of the screen, occupying an area with diameter X_{human}, and that the space around the user X_{frame} is enough so that the option buttons can be displayed. X_{human} and X_{frame} are directly proportional to the angles γ and ψ, respectively, which accumulates to the viewing angle of the camera, θ. As the height H_{sh} and width W_{sh} of the user's shoulders are fixed for a specific user, the desired input image can be obtained by either adjusting the height of the camera H_{cam} and keeping the viewing angle of the camera θ fixed, or by changing the viewing angle θ and keeping the height of the camera fixed. This means that the system can be set up in a place with a low or high ceiling, with the ceiling height depending on the viewing angle θ of the camera and visa versa.

This dependency is shown in the two setup equations shown below that was derived in a previous paper by the authors [10]. The derivations made use of the golden ratio, φ, as well as the fact that the angle ψ is a factor δ of the viewing angle θ.

$$\theta = \frac{2}{(1-2\delta)} \arctan\left(\frac{H_{human}}{2(2\varphi+1)(H_{cam}-H_{sh})}\right) \quad (1)$$

$$H_{cam} = \left(\frac{H_{human}}{2(2\varphi+1)\tan\left(\frac{(1-2\delta)\theta}{2}\right)}\right) + H_{sh} \quad (2)$$

where

$$H_{sh} = H_{human}\left(\frac{39\varphi+17}{46\varphi+19}\right) \quad (3)$$

5. Implementation

HandsFree was implemented in C++ with the image processing being done on a NVIDIA GeForce 7300 graphics card. A dependency diagram of the implemented system is shown in Fig. 7. The subsystems that formed part of the image processing, starting at the foundation, are as follow:

- Graphics Processing Unit (GPU)

The graphics processor has many processors in parallel (the amount being different for different types of graphics processors) which gives it the ability to process images much faster than a CPU. The image processing for this system is done on the GPU, although the rest of the processing is done on the CPU. How it was done will be explained later on.

- Simple DirectMedia Layer (SDL)

SDL is a dynamically linked library that is used but that was not written by the author. SDL was used for its window handling and timer capabilities. During development keyboard and mouse input was used and SDL also provided the handling of these input devices.

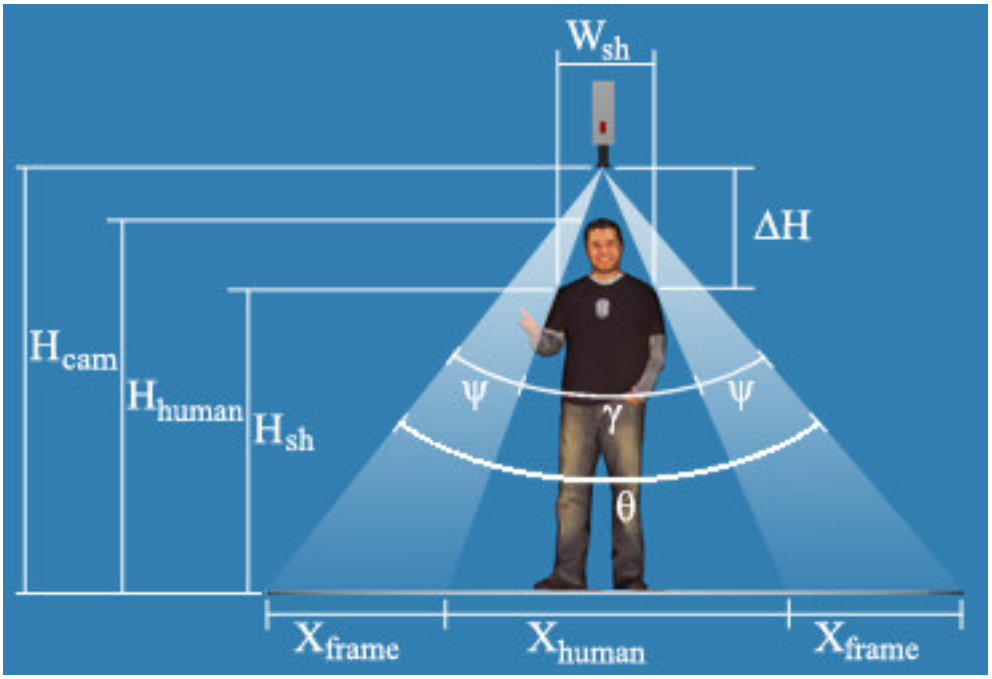


Figure 6: *Camera setup*

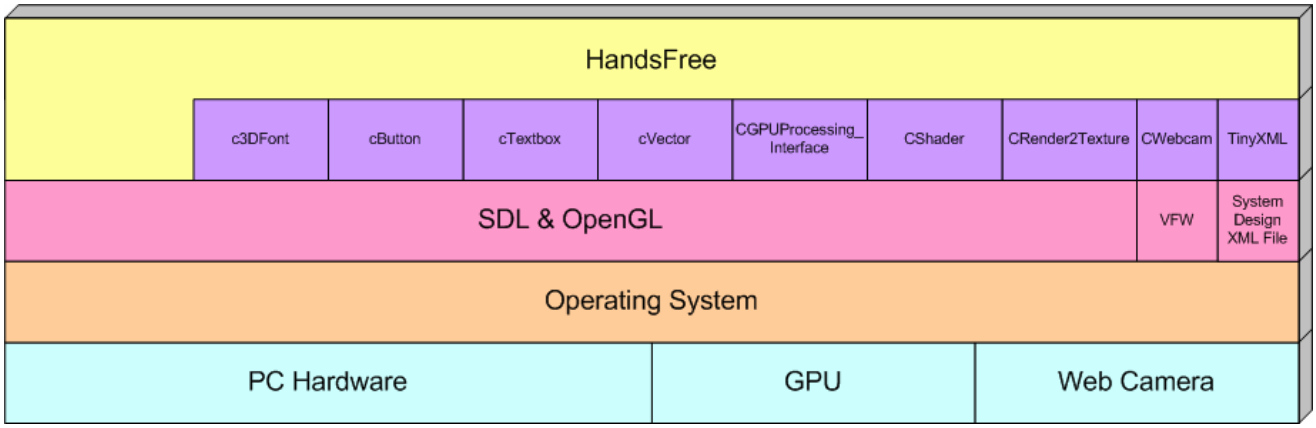


Figure 7: *Dependency diagram of the marker-free VBI device*

- OpenGL

OpenGL is an interface between the CPU and the GPU. It can be seen as a set of built in functions to manipulate 3D objects in a 3D environment as well as the objects' colours and textures. The output of the system is displayed in a 3D environment on the screen. The output image is a texture that is placed on a rectangle, or quad as it is referred to in OpenGL. This quad can be placed anywhere and at any orientation in the 3D environment, but for this project it was decided to make the quad the same size as the screen, with the orientation being the same direction as the screen, in other words on the z axis.

- Video For Windows (VFW)

This is used to interact with streaming video and is used by the CWebcam class to interact with the webcam.

- c3DFont

As the output exist in a 3D environment, text can not be displayed in the same fashion as it would have been when for instance a DOS box was used. c3DFont is a class that provides the functionality of displaying text in a 3D environment. It takes ASCII text as an input, together with the size the text should be, and constructs 3D shapes of the text which can then be rendered in the 3D environment on the screen.

- CGPUProcessing_Interface

There are a few standard functions that are often used in image processing such as changing a colour image to its grey-level equivalent or to a duotone black and white image. This set, which is still growing, has been brought together in the CGPUProcessing_Interface class. The functions of this class that are used in this project are the bgra2rgba function that swaps the blue and red channels of the incoming image and the vertical flip function that flips the image vertically. This is necessary as the webcam does not send the captured image in a RGB format with the first elements in the array belonging to the top left corner, which is generally the custom.

- CRender2Texture

This class allows a texture to be rendered onto a virtual quad, meaning one that is not necessarily displayed. This quad can be smaller than the original image. If for instance a 128x128 image is rendered onto a 32x32 quad, then every block of 4x4 pixels on the image is averaged to produce a pixel on the quad. This quad can then again be stretched to produce a 128x128 averaged image. This technique is used to do averaging as it requires less processing than a written function that does the same. The reason averaging is used in this application is to determine the background's average for comparison to see if users are holding their hands over a button. The reason the image of the background is not used as is, is because on a per pixel basis a pixel at a specific location can change a lot due to small vibrations on location and

due to non-perfect image capturing technology being used by the camera.

- CShader

A shader program is one that is performed either per geometrical shape, per vertex or per pixel and is done on the GPU. These shader programmes can be written in Assembly language but are more generally done in the C++ language. This system made use of pixel shaders to do the image processing and was written in the C++ language. These shaders are stored in a text file and the CShader class is used to load these programmes into the GPU and to execute them on each pixel in the image. There were two shaders that were written for this system. The one shader mixes the current averaged image that was rendered by CRender2Texture with the average history texture. By making the mix percentage very low, the average history texture is able to change over a long time without the user's moving hand affecting the background a lot. This continual learning is important as the background can change during the day due to light changes on the location. The other shader determines how many pixels are different from the background and makes use of the average texture, the current image and a black-and-white map of the buttons' locations. This shader is used to determine which button the user is currently selecting, if any.

- CWebcam

CWebcam, together with VFW, gives the system access to the images that is sent from the webcam. In earlier versions of the system when the processing was being done on the CPU, this class grabbed an image and placed it in an array. With the processing now being done on the GPU, an image is grabbed and placed directly in the GPU's memory, thus increasing the overall processing speed of the system.

- HandsFree

HandsFree is the overhead class that ties all the above mentioned subsystems together. This class is also responsible for determining which button is pressed, i.e. which button is covered most by the user's hands. It also reads the XML file and calls the appropriate DLL or web service when it is required.

6. Experimental Setup

The usability and restrictions of HandsFree depends on its pre-defined colour range and selection time, as mentioned in section 3. The aim of the three experiments was to find the ideal settings for these two parameters. The first two experiments set out to find the ideal colour range.

HandsFree is intended to work in an array of environments, but like any system it has its limitations. The first experiment looked at different background colours to see if the difference between the skin of a Caucasian hand and the background could be detected, i.e. if the user was able to select and press a button for a certain background. In this experiment the light was kept constant. A set of 35 colours from the Plascon colour range was chosen as it can easily be attained by the end user. The colour set that was chosen for testing was series 06 and is between yellow and orange. The colours in the series vary from an orange to a dark brown. Series 06 was chosen as it includes most skin shades. The series is divided into five shades with seven colours in each shade. These 35 colours are given in Fig. 8.

Figure 8: *Plascon Series 06 colours*

Three different Colour Ranges (CR) were tested, namely 0.02, 0.05 and 0.07. Background colour is given as three values, each between 0 and 1, representing the amount of red, green and blue in the colour. If the background colour is for example R=0.7, G=0.5 and B=0.6 and a colour range of 0.02 was chosen, then colours that has red values between 0.68-0.72, green values between 0.48-0.52 as well as blue values between 0.58-0.62 would be seen as background colours. Thus the larger the colour range, the more colours are included in the background-colour-range and the more difficult it is to detect skin colour. The test was performed three times, once for each of the colour range values, on each of the 35 colours.

Experiment 2 is the inverse of the first experiment. Here the background colour was kept constant with the light source being varied. The three colour ranges were tested for each of eight different Light Source Levels (LSL) ranging non-linearly from 2.5 Lux to 10240 Lux, with the result (indicating if the user could select and press a button or not) recorded each time.

The time a button is selected before it is pressed, i.e. the selection time, determines the responsiveness of the system and should be set by users according to what they feel is appropriate to them. This may vary from someone who moves slower to others who prefer the system to have a quick response. But there is another factor that comes into play when setting the selection time of HandsFree. This is the factor of windows in the room together with the sunlight on a partly cloudy day.

HandsFree does continual learning of the background. This enables the system to track changes in the background. The learning rate is 10 seconds which means that a change will take 10 seconds to be fully incorporated into the learned background. This slow learning enables users to select a button while subtle light changes in the room will not do the same. But on a partly cloudy day clouds can move in front of or away from the sun causing sudden light changes.

In Experiment 3 the light intensity is changed from LSL 3 (200 Lux) to LSL 4 (320 Lux) in different time periods starting at 10 seconds down to 0.1 seconds. This is done for different selection times starting at 0.2 seconds with increments of 0.2 seconds up to 1.2 seconds. In each case it was recorded if the system automatically selected and pressed a button due to the light change.

All three experiments were done on two different high quality web cameras in order to compare the results. The one is the A4 Tech PK-336MB web camera and the other the Canyon CNR-WCAM413. The results of each experiment are presented as well as discussed in the following section.

7. Experimental Results and Discussion

7.1. Experiment 1

In all the tables to follow "Y" indicates that the button was selected and pressed by the user's hand, "N" indicates that the user was not able to press the button and "S" indicates that the button was automatically selected and pressed without user input. This phenomenon of automatic pressing has to do with the camera's automatic light adjustment when viewing dark colours. The phenomenon is caused by enough of these adjusted pixels falling outside the colour range to indicate a button as selected. A continuation of this phenomenon will cause the button to be automatically pressed.

Table 1: *CR = 0.02; A4 Tech*

	1-1	1-2	1-3	1-4	2-1	2-2	2-3
A	Y	Y	Y	Y	Y	Y	Y
B	Y	Y	Y	Y	Y	Y	Y
C	Y	Y	Y	Y	Y	Y	Y
D	Y	Y	Y	Y	Y	Y	Y
E	S	S	Y	Y	Y	Y	Y

Table 2: *CR = 0.02; Canyon*

	1-1	1-2	1-3	1-4	2-1	2-2	2-3
A	Y	Y	Y	Y	Y	Y	Y
B	Y	Y	Y	Y	Y	Y	Y
C	Y	Y	Y	Y	Y	Y	Y
D	S	S	Y	Y	Y	Y	Y
E	S	S	S	Y	Y	Y	Y

Table 3: *CR = 0.05; A4 Tech*

	1-1	1-2	1-3	1-4	2-1	2-2	2-3
A	Y	Y	N	N	Y	Y	Y
B	Y	Y	N	Y	Y	Y	Y
C	Y	Y	N	Y	Y	Y	Y
D	Y	N	Y	Y	Y	Y	Y
E	Y	Y	Y	Y	Y	Y	Y

Table 4: *CR = 0.05; Canyon*

	1-1	1-2	1-3	1-4	2-1	2-2	2-3
A	Y	Y	Y	Y	Y	Y	Y
B	Y	Y	Y	Y	Y	Y	Y
C	Y	N	N	N	Y	Y	Y
D	Y	Y	Y	Y	Y	Y	Y
E	Y	Y	Y	Y	Y	Y	Y

Table 5: *CR = 0.07; A4 Tech*

	1-1	1-2	1-3	1-4	2-1	2-2	2-3
A	Y	N	N	N	N	Y	Y
B	Y	N	N	N	Y	Y	Y
C	Y	N	N	N	Y	Y	Y
D	N	N	N	Y	Y	Y	Y
E	N	N	Y	Y	Y	Y	Y

Table 6: *CR = 0.07; Canyon*

	1-1	1-2	1-3	1-4	2-1	2-2	2-3
A	Y	N	N	N	N	Y	Y
B	N	N	N	N	N	Y	Y
C	Y	N	N	N	N	Y	Y
D	N	N	N	N	N	Y	Y
E	Y	Y	N	Y	Y	Y	Y

As the colours in Fig. 7 goes from one extreme to the next it was expected that similarities between the Caucasian hand and the background would be found somewhere in the middle. This was the case with a colour range of 0.05 and 0.07. In the latter there are too many colours that do not cause the Caucasian hand to trigger a button. In the case with a colour range of 0.02 there are no colours where the hand does not trigger a button. At first glance this setting seems ideal, but there are colours that cause buttons to be automatically pressed. When one of these colours are in the background and do cause a button to be pressed without the user's consent, it becomes very difficult to operate the system. With a colour range of 0.02 being too responsive and one of 0.07 being somewhat unresponsive, it seems that working with a colour range of 0.05 give the best, although not perfect, response.

7.2. Experiment 2

Table 7: *Colour range investigation in different light intensities using an A4 Tech web camera*

	LSL 0	LSL1	LSL2	LSL3	LSL4	LSL5	LSL6	LSL7	LSL8
EV	0.0	1.7	5.4	6.3	7.0	8.5	10.8	11.5	12.0
Lux	2.5	8.1	105.6	197.0	320.0	905.1	4457	7241	10240
Image									
CR = 0.02	S	S	Y	Y	Y	Y	Y	Y	Y
CR = 0.05	S	Y	N	N	Y	Y	Y	Y	N
CR = 0.07	Y	N	N	N	N	Y	Y	N	N

Table 8: *Colour range investigation in different light intensities using a Canyon web camera*

	LSL 0	LSL1	LSL2	LSL3	LSL4	LSL5	LSL6	LSL7	LSL8
EV	0.0	1.7	5.4	6.3	7.0	8.5	10.8	11.5	12.0
Lux	2.5	8.1	105.6	197.0	320.0	905.1	4457	7241	10240
Image									
CR = 0.02	S	S	Y	Y	Y	S	Y	Y	Y
CR = 0.05	S	S	Y	Y	Y	Y	Y	Y	Y
CR = 0.07	S	S	Y	Y	Y	Y	Y	Y	Y

From the above results there are a few observations that can be made. First it is seen that different web cameras have different algorithms for performing automatic light adjustment. Due to these varying adjustment methods, different cameras are able to work in different light ranges making some cameras more robust in this area. Finally, as in the previous experiment, a colour range of 0.05 performs well when using a better quality camera.

7.3. Experiment 3

Table 9: *Light change triggering with an A4 Tech web Camera*

Time (sec)	Speed (Lux/sec)	0.2 sec	0.4 sec	0.6 sec	0.8 sec	0.12 sec
10	12.00	N	N	N	N	N
9	13.33	N	N	N	N	N
8	15.00	N	N	N	N	N
7	17.14	N	N	N	N	N
6	20.00	N	N	N	N	N
5	24.00	N	N	N	N	N
4	30.00	N	N	N	N	N
3	40.00	N	N	N	N	N
2	60.00	N	N	N	N	N
1	120.00	N	N	N	N	N
0.5	240.00	Y	Y	N	N	N
0.1	1200.00	Y	Y	N	N	N

Table 10: *Light change triggering with a Canyon web Camera*

Time (sec)	Speed (Lux/sec)	0.2 sec	0.4 sec	0.6 sec	0.8 sec	0.12 sec
10	12.00	N	N	N	N	N
9	13.33	N	N	N	N	N
8	15.00	N	N	N	N	N
7	17.14	N	N	N	N	N
6	20.00	N	N	N	N	N
5	24.00	N	N	N	N	N
4	30.00	N	N	N	N	N
3	40.00	N	N	N	N	N
2	60.00	N	N	N	N	N
1	120.00	N	N	N	N	N
0.5	240.00	N	N	N	N	N
0.1	1200.00	Y	Y	N	N	N

Tables 9 and 10 shows that sudden change in light, like when a cloud moves in front of the sun, has no effect on HandsFree if a selection time of 0.6 seconds or more is chosen.

8. Conclusion

In this paper a marker-free VBI device for the use with menu-driven systems was proposed. By implementing the image processing parts on the graphics card using Shader technology, it is possible for the system to run in real-time.

The system was tested against background colours that are close to that of Caucasian skin. The results showed that using a colour range of 0.05, the system works with the most colours tested without the system automatically pressing buttons. This setting of 0.05 also showed good results in different lighting intensities. HandsFree was also shown to be robust against sudden light changes if a selection time of 0.6 or more is used. This is important as the system must be able to work in surroundings where daylight is present on days when the weather is partly cloudy.

References

[1] Remondino, F., "Tracking human movements in image space", Internal technical report at IGP - ETH Zurich, 2001

[2] SCEE Limited, "What Is EyeToy?", http://www.eyetoy.com/index.asp?pageID=18, Referenced on 1 October 2007

[3] Hansen, D. W. and Hammoud, R. I., "An improved likelihood model for eye tracking", Computer Vision and Image Understanding, 2007

[4] Manresa-Yee, C., Varona, X. and Perales López, F. J., "Towards Hands-Free Interfaces Based on Real-Time Robust Facial Gesture Recognition", AMDO, 2006, pp.504-513

[5] Kölsch, M. and Turk, M., "Fast 2D Hand Tracking with Flocks of Features and Multi-Cue Integration", IEEE Workshop on Real-Time Vision for Human-Computer Interaction (at CVPR), 2004

[6] Störring, M., Andersen, H. J. and Granum, E., "Physics-based Modelling of Human Skin Colour under Mixed Illuminants", Robotics and Autonomous Systems, 35(3-4), 2001, pp. 131-142

[7] Heidemann, G. Bax, I. and Bekel, H., "Multimodal Interaction in an Augmented Reality Scenario", ICMI'04, 2004, pp. 1-8

[8] Devlaeminck, R., "Human Motion Tracking with Multiple Cameras Using a Probabilistic Framework for Posture Estimation", Masters Degree in Electrical and Computer Engineering, Purdue University, Indiana, 2006

[9] Han, J. and Bhanu, B., "Fusion of Colour and Infrared Video for Moving Human Detection", Pattern Recognition, 2006

[10] Visser, W., Roodt, Y. and Clarke, W. A., "Design Considerations for a Marker-free Visual-Based Interfacing Device for Telco Operation", SATNAC, 2007