Interactive Whiteboard JAVA API

Andrej FOGELTON*

Slovak University of Technology
Faculty of Informatics and Information Technologies
Ilkovičova 3, 842 16 Bratislava, Slovakia
fogelton@gmail.com

Abstract. Although, interactive whiteboards are very useful they are not very common in the teaching process due to their high price. The purpose of our work is to create a very cheap software solution using a webcam. It is based on tracking light from a pen, which will have a light-emitting diode (LED) attached at the top. Illuminated spot will be detected by image processing methods. Location of this point will be converted into the coordinate system of the screen by the principle of homography.

In the future we would like to develop JAVA API that would provide events as the left mouse click or move of this pen. As a demo application (prototype) we used our framework to control the computer mouse by our pen.

1 Introduction

Interactive whiteboards are very popular among teachers and students, because they are simple to use and also more comfortable than whiteboards. They are not only comfortable but very practical and time saving. You can save your work to the computer, but they are not used very often. The main reason is their high price, however there is one cheap solution. It is based on using Nintendo Wii Remote as an infrared camera which is pointing on the board with a modified pen [2]. This pen has an infrared diode attached on the top. It is a great solution, but Nintendo Wii Remote is not very common among people. That is why we decided to create a software solution, which will be processing the images from a webcam. Webcam is a very common hardware among users and is much cheaper than Nintendo Wii Remote.

* Bachelor study programme in field: Informatics
Supervisor: Ing. Matej Makula, Institute of Applied Informatics, Faculty of Informatics and Information Technologies STU in Bratislava

The first idea was to use a webcam as an infrared capturing device by setting the driver to detect infrared light only. Later, by improving the prototype we found out that we would be wasting the abilities of the webcam. For example the ability to detect different colours is used to calibrate the software properly. The only thing that the user has to do is to place a webcam properly and run calibration process. A homemade pen has a switch and a red LED at the top and it is very cheap to make. Our solution can be used either with LCD monitor or an projector. There are some solutions with similar principles.

Touchlib [6] is a library for creating multi-touch interaction surfaces. It handles tracking blobs of infrared light, and sends these multi-touch events to your programs, such as ”finger down”, ”finger moved”, and ”finger released”. It includes a configuration application and a few demos to get you started, and will interface with most types of webcams and video capture devices. It currently works only under Windows but efforts are being made to port it to other platforms.

Another software Laser Pointer Map Interaction [3] creates a prototype of laser pointer recognition system for geography lessons at high schools. It is based on tracking the position of laser point on information table (blind map of Slovakia in our case) using an ordinary webcam. By processing the webcam input we get the exact position on the map and the name of the nearest known city is played as a sound from our sound records.

2 Main hardware and software principles

Our prototype is programmed in JAVA using Java Media Framework [4], which provides access the webcam. As a pointing device a homemade pen with LED at the top is used (Figure 1). This LED is switched on by pressing a very sensitive switch, which was reused from an old floppy drive. The reason why we used a very sensitive switch was to minimize the change of the pen location when pressing the switch. Any arbitrary webcam with minimal 1.3 Mpix resolutions can be used as a capturing device. Smaller resolution could cause bigger inaccuracy. Before starting our prototype up, it is desirable to set up the webcam driver properly (Table 1). These settings are the result of continuous
testing during development process. They enable work in most luminous conditions, however there still might be some inaccuracy caused by sharp sun light.

2.1 Location of the webcam

Before being able to use this interactive method, the webcam has to be located to view the whole pointing area (LCD or whiteboard). The distance between the webcam and the display surface depends on the viewing angle of the webcam, which is normally about 50 degrees. Most of the tests were performed on LCD displays and from these tests the following results were concluded.

The optimal location is directed straight to the screen, but this location is not very comfortable during work, because the webcam is hindering. From these tests we know that the maximum angle from the middle of the screen is about 30 degrees. It is due to a bad viewing angle of the normal LCD display. With growing angle, the precision of the application software reduces.

The location of the webcam depends on whether the user is left or right handed. Otherwise, the light from the pen could be shaded by his hand. See the Figure 2 below.

![Figure 2. Location of the webcam for a right handed person](image-url)
2.2 Self-calibration

At the first stage, the prototype has to be calibrated. The calibration is inevitable to find out the location and size of the LCD or whiteboard. It is provided by a red calibration screen in which blue calibration spots are moving (Figure 3). Processing of two images is made to find out the location of the calibration spot. Colours of the first image (red screen) are compared to the second (red screen containing a blue calibration spot):

\[ |r_1 - r_2| + |b_1 - b_2| > \tau, \]

where \( r_1 \) is the red element of some point in first image, \( b_2 \) is the blue element of some point in the second image and \( \tau \) is the threshold variable used for detecting sufficient colour change. We found out that the optimal value is about 10% of maximum intensity, depending on light conditions.

If this condition is satisfied at some point, location of this point is remembered and at the end the arithmetic average of these points is calculated. This procedure is repeated for each calibration spot. These identified points are in the coordinate system of the webcam, not the screen. The homography matrix has to be computed in order to transform points from the webcam coordinate system to the coordinate system of the display surface.

2.3 The principle of homography

Homography [1] is a concept in the mathematical science of geometry. A homography is an invertible transformation from a projective plane to a projective plane that maps straight lines to straight lines. Synonyms are collineation, projective transformation, and projectivity. Figure 4 shows, that the distances and locations in the pictures are different from reality. Variables were defined to transform individual points from pictures to their real actual location.

To transform webcam coordinates \( x, y \) to screen coordinates \( \hat{x}, \hat{y} \), we will need the
Writing the $H$ matrix in vector form as $h = (h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32}, h_{33})^T$ the homogeneous equations for $n$ points become $|A \cdot h| = 0$ with $A$ the $2n \times 9$ matrix. Values of $x$ and $y$ are known, but the $H$ matrix is unknown. This is computed by the following formula $|A \cdot h| = 0$, where the $A$ matrix is defined as:

$$ A = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1 \hat{x}_1 & -y_1 \hat{x}_1 & -\hat{x}_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1 \hat{y}_1 & -y_1 \hat{y}_1 & -\hat{y}_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2 \hat{x}_2 & -y_2 \hat{x}_2 & -\hat{x}_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2 \hat{y}_2 & -y_2 \hat{y}_2 & -\hat{y}_2 \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_n \hat{x}_n & -y_n \hat{x}_n & -\hat{x}_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -x_n \hat{y}_n & -y_n \hat{y}_n & -\hat{y}_n \end{bmatrix} \quad (3) $$

It is a standard result of linear algebra that the vector $h$ that minimises the algebraic residuals $|A \cdot h|$, subject to $|h| = 1$, is given by the eigenvector of least eigenvalue of $A^T A$. This eigenvector can be obtained directly from the singular value decomposition of $A$ [5]. In the case of $n = 4$, is the null-vector of $A$ and the residuals are zero.

$$ A_{2n \times 9} = U_{2n \times 2n} S_{2n \times 9} V_{9 \times 9}^T \quad (4) $$

Figure 4. The principle of homography
\[ V^T V = I \]  
\[ U^T U = I \]

The solution for our case will be the row of the matrix \( V \). The number of the row is determined by the number of the row of the \( S \) matrix with the lowest value. This row will be divided into 3 rows by 3 numbers to create the \( H \) matrix.

3 Tracking the LED pen

After self-calibration, the software is tracking the light of the LED pen. Each image is processed to find out the high intensity of light emitted by LED. Each RGB colour point contains red \( r \), green \( g \) and blue \( b \) colour elements. For the points satisfying condition:

\[ \frac{(r + g + b)}{3} > \tau, \]

the arithmetic average of position is calculated. This point, which is in the coordinate system of the webcam, is transformed to the coordinate system of the display device by multiplication with the \( H \) matrix.

In calibration mode it can be checked how much of the image satisfies the condition and from how many points the average is calculated. If there is a lot of tracked light or nothing the value of \( \tau \) has to be changed to the proper value for tracking the optimal amount of light.

3.1 Controlling the mouse pointer

After detecting the high intensity of light, the coordinates of the light are computed. In case there was no sharp light in the previous image, software emulates the left click of the mouse. During a continuous and uninterrupted detection, the cursor of the mouse follows the light. When no light is detected, a signal to release left click is sent out.

This software solution requires a webcam with minimal resolution 640 x 480 with 30 fps, due to the fluent move of the mouse. This requires a lot of computing power.

4 PenListener JAVA API

The result of the prototype is a new interface of Listener. This interface provides the ability for using the following events.

```java
public void penPressed(PenEvent e) {
    System.out.printf("Pen pressed", e);
}

public void penReleased(PenEvent e) {
    System.out.printf("Pen released", e);
}
```
public void penMoved(PenEvent e) {
    System.out.printf("Pen moved", e);
}

5 Conclusion

The results of alfa testing of the prototype showed up that there is a good aim, but there are still some unsolved issues, for example high computing severity or extension of our prototype. There is an assumption that due to low price of this technology, developers can use this API in high level application for interactive whiteboards. The real advantage of this is that programmers do not have to program the basic operation by their own. But they can simply use this API and concentrate on the final user application. This JAVA API can be later extended by adding right mouse click, which could be realized by LED of different colour or multicolour LED.

References


