Volume 3, Issue 1, January 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Laser Powered Virtual Events

Manjunatha V*
Department of Computer Science & Engineering ,
East West Institute of Technology, VTU
India

Prasanna Kumar M
Department of Computer Science & Engineering ,
East West Institute of Technology, VTU
India

Abstract— ***with the advancement of technology devices are becoming affordable to general people and people are more attracted towards Artificial Intelligence and new ways to interact with computer devices. Laser Powered Virtual Events is a java based application that detects the laser point stroke on any flat surface which is divided into 9 segments. Here we can make any flat surface such as wall or book as virtual events and a press of laser light acts as key press on keyboard. Laser Powered Virtual Events uses webcam to monitor the segmentation area, detects if any laser light comes to any segment and then does the equivalent of pressing a specified keyboard key. Here to interact with media device i.e. camera we use JMF (Java Media Framework).***

Keywords— Laser Point, Key Board, Automatic Events, Artificial Intelligence

I. INTRODUCTION

Laser Control is an application I wrote to control keyboard using a laser pointer by pointing defined areas on a wall. This application monitors the wall for the presence of a small spot using a webcam. When the spot appears somewhere, if it is inside a defined hotspot, the appropriate command or keystroke is executed. Laser Control is a java program that using a webcam detects when a laser pointer is in a specified hotspot and then does the equivalent of pressing a specified keyboard key. Thus we can control Media Portal or applications through the hot grids. The hotspots can be set by the user as well as the actions taken. We can make our wall as a remote control.

For example I like to music on my laptop, but changing tracks is a real pain since I have to reach for the keyboard every time I feel like changing a song (and also because I'm very lazy)! Well, as the old saying goes, "laziness is the seed from which inventions flower". Even though I just made that up, it's no less true! Anyway, I thought that it would be great if I could somehow give visual "gesture" commands to my laptop and control Windows Media Player without having to touch the keyboard. So, I fired up my code editor and wrote a laser point gesture recognition program which currently recognizes up to Nine visual gestures (left, right, up, down and other 5 keys of my choice which may change anytime). Fig 1 shows a webcam view from the program and key strokes press with a laser beam. However, the problem with creating hotspots is that we have to remember grids and their keys. If the room's dark, we have a really hard time finding and activating the key strokes with laser. However, with gesture recognition, all we have to do is make a gesture anywhere in the camera's field of view and that's it!

The diagram below shows how projecting a laser dot onto a target that is in the field of view of a camera.

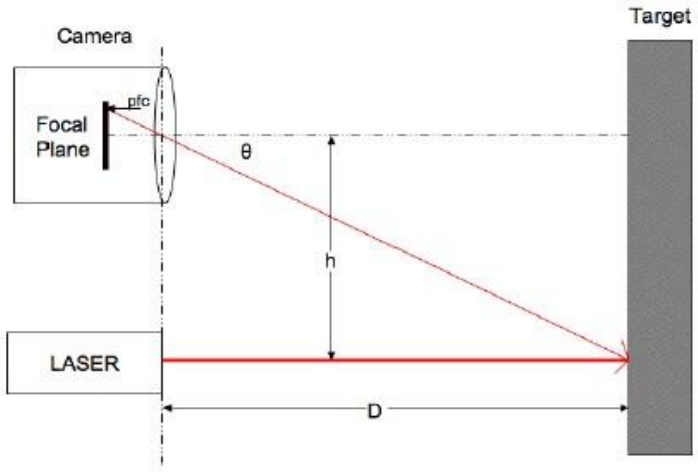

Fig.1 projecting a laser dot onto a target

So, here is how it works. A laser-beam is projected onto an object in the field of view of a camera. This laser beam is ideally parallel to the optical axis of the camera. The dot from the laser is captured along with the rest of the scene by the camera. A simple algorithm is run over the image looking for the brightest pixels. Assuming that the laser is the brightest area of the scene (which seems to be true for my dollar store laser pointer indoors), the dots position in the image frame is known.

The reflected dot loses intensity as it bounces within the camera so it does not interfere with the algorithm that detects the brightest pixel in the image.

A. Objective of the Study

- Our system would be capable of replacing the generating events by laser point on the wall or any flat surface as a new way to interact with the computer.
- Laser Pointer is used to interact with computer.
- The Laser Light stroke on the surface would be captured by camera and taken as key press event.

B. Scope

In our system, we divide the any flat surface which is being captured by camera, into 9 grids and assign keyboard keys to them. So every time we want to assign different key to grids we need to make change in the configuration file and restart the system.

II. SYSTEM ANALYSIS

A. Existing System

In existing system people use only computers to operate it and do any work. Even at some high tech places we can see laser keyboard which is expensive and out of reach for common people. Here the major drawback with all the types is people need to they need to keep the keyboard near to them and press the keys manually.

B. Proposed System

We propose a system where remotely a person could be able to access the computer. To make the system affordable to general public we would make use of low price webcam and a laser point to for operation. Our system would be capable to map any flat surface as virtual keypad and translate laser stoke as key press.

C. System Specification

1) Hardware Requirement(Minimum Requirement)

Processor : Intel Dual Core

RAM : 256 MB

Hard Disk: 40 GB

Camera : Web Camera (RGB Supported)

Laser Pointer : .5mm laser pointer

2) Software Requirement

Technology: JAVA

Tools: Java 1.6, java Media frame Work, JMS

Platform: Windows 98/2000

E. System Architecture

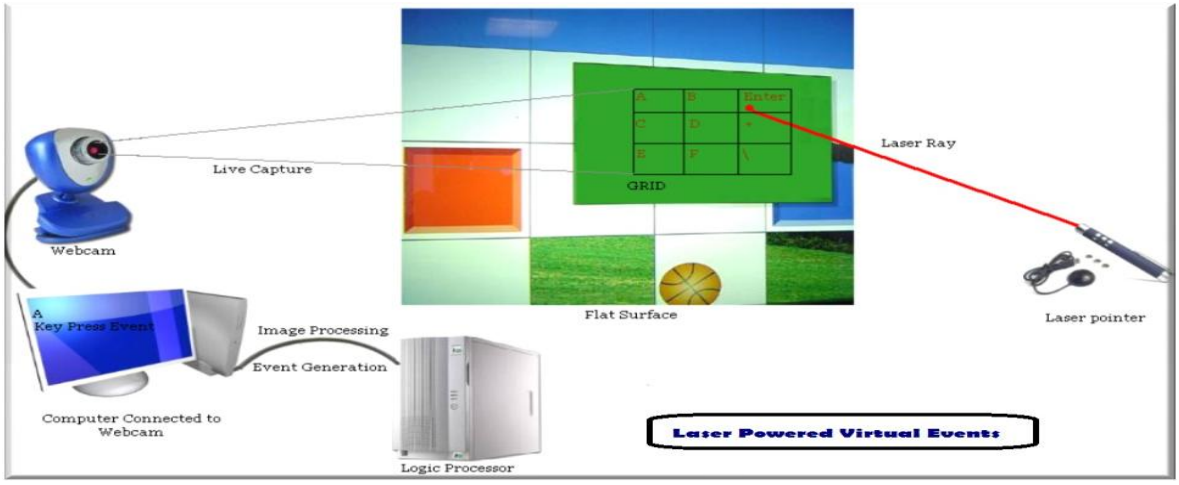

Fig.2 System Architecture

III. MODULE DESCRIPTION

A. Webcam Capture Module

In this module we configure JMF (Java Media Framework) to the application and write code for interaction with webcam. Our application does live capturing of the ongoing video and display it in the application frame. We can set the attributes for the camera and resolution settings through command line arguments or it would take directly from the video Properties file.

B. Grid Creation Module

In this module we divide the frames in grids by providing size and dimensions. These grids act as keys to fire the event. If we point laser on any of the grid then the associated event will be fired.

C. Laser Point Identifier Module

Here a fixed camera captures a scene which is analysed as frames. Every time next frame gets compared with the previous frame for the changes. In this module we write program for pixel colour change detection in frames. Here we

analyse the base frame pixel colours and match with the next frame pixels. Because of the brightness of laser point it gets detect by our application.

D. Anti-Repeat Module

To prevent a hotspot from being activated more than once we implement anti repeat system which will avoid the repetition of clicks because of minor point movements.

E. JDBC & Database Module

In this module we store each activity done by the user in database. For this purpose we are using JDBC Type 1 driver. We need to create the database in SQL Server and DSN as "laser-grid". We store the Grid name, date/time of the event and the event get fired because of Laser clicks.

IV. SYSTEM IMPLEMENTATION

Pictures are continuously received from the webcam. Here basically we put an infinite loop for capturing the frame and put image in buffer so that we can compare it with the next frame image. A very simple method to track the laser point is to recognize the part where the frame difference is found. So we divide the entire surface into 9 Grids and look for the changed coordinate within all the grids. All the grids are pre specified for their own functions so whichever grid contains the changes picture the respective function is called.

The rectangle containing all the pixels that have changed is calculated. Since the laser only generates a small spot, we check the size of the rectangle:

If the rectangle is too big, it is ignored. If the rectangle is very big, our application assumes the difference is due to a light that has just been turned on or off. To prevent the light event to trigger hotspots by mistake, we keep 1/4 second of gap between the capturing of images (configurable) received from the webcam and the unintended strokes are ignored. This allows the webcam to adjust itself to the new laser stroke and execute event as per it.

If the rectangle size makes sense, the list of hotspots is searched for an hotspot in which the rectangle fits.

A. Anti-repeat

To prevent a hotspot from being activated more than once because you left the pointer inside it a little too long, the following strategies are used: Once a hotspot is activated, the following pictures containing changes (valid or not) are ignored until a picture without changes (not laser pointer) is received. (Note: This does not apply to repeatable zones) When a hotpot is activated and the hotspot is disabled during one second. Take the camera, point it at something. Shine the laser inside the area where the camera can see. That's it in a nutshell. However, there are some additional considerations.

First, the more direct the camera, the more accurate it will be. If the camera is off to the side, it will see a skewed wall, and because one side is closer than the other, it's impossible to focus perfectly, and one side of the wall will be more precise than the far side of the wall. Having the camera point as directly as possible and at the hooked surface is the best option.

Second, the distance to the surface matters. A camera that is too far from the surface may not be able to see a really small laser point. A camera that is too close will see a very large laser point and will not have great resolution. Its tradeoffs and you just have to experiment to find the best distance.

Third, if using wall as the projected area, the camera should be able to see slightly more than the projected area. A border of a few inches up to a foot is preferable, as this space can actually be used for input even if it's not in the projected space.

Fourth, it's important to control light levels. If there are sources of light in the view of the camera, such as a lamp or window, then it is very likely the algorithm will see those points as above the threshold, and will try to consider them part of the laser pointer (remember white light is made up of red, green, and blue, so it will still be above the red threshold). Also, if using a projector, the laser pointer has to be brighter than the brightest the projector can get, and the threshold has to be set so that the projector itself isn't bright enough to go over the threshold. And the ambient light in the room can't be so bright that the threshold has to be really high and thus the laser pointer isn't recognized. Again, there are a lot of tradeoffs with the light levels in a room.

I wrote my software in java, there are two libraries that depend heavily on:
http://java.sun.com/javase/technologies/desktop/media/jmd/2.1.1/download.html java media framework (JMF) 2.1.1e
this is used for capturing the individual frames of the camera.

B. Basic Theory

The basic theory behind the project is the following; a laser pointer shines on a surface. The web camera is looking at that surface. Software running on a computer is analysing each frame of that camera image and looking for the laser pointer. Once it finds that pointer, it converts the camera coordinates of that point into screen coordinates and fires an event to any piece of software that is listening. That listening software can then do something with that event. The simplest example is a mouse emulator, which merely moves the mouse to the correct coordinates based on the location of the laser.

C. *Implementation Theory*

To implement this, I have the JMF library looking at each frame. I used this frame access.java example code as a starting point. When looking at each frame, I only look at the 320×240 frame, and specifically only at the red value.

Each pixel has a value for red, green, and blue, but since this is a red laser I'm looking at, I don't really care about anything but red. I traverse the entire frame and create a list of any pixels above a certain threshold value. These are the brightest pixels in the frame and very likely a laser pointer. Then I average the locations of these points and come up with a single number. This is very important, and I'll describe some of the effects that this has later. I take this point and perform the affine transform to convert it to screen coordinates. Then I have certain events that get fired depending on some specific things:

- Laser On: the laser is now visible but wasn't before.
- Laser Off: the laser is no longer visible.
- Laser Stable: the laser is on but hasn't moved.
- Laser Moved: the laser has changed location.
- Laser Entered Space: the laser has entered the coordinate space (I'll explain this later).
- Laser Exited Space: the laser is still visible, but it no longer maps inside the coordinate space.

For most of these events, the raw camera coordinates and the transformed coordinates are passed to the listeners. The listeners then do whatever they want with this information.

Limitations:

For most of these events, the raw camera coordinates and the transformed coordinates are passed to the listeners. The listeners then do whatever they want with this information.

V. CONCLUSION

Laser Powered Virtual Keyboard uses webcam to monitor the segmentation area, detects if any laser light comes to any segment and then does the equivalent of pressing a specified keyboard key. Here to interact with media device i.e. camera we use JMF (Java Media Framework). Laser Control is a java program that using a webcam detects when a laser pointer is in a specified hotspot and then does the equivalent of pressing a specified keyboard key. Thus we can control Media Portal or applications through the hot grids. The hotspots can be set by the user as well as the actions taken. We can make our wall as a remote control.

VI. FUTURE WORK

We can enhance the system to take complete keyboard keys in form of grids and at the time of projector presentation, it can be used for changing slides remotely or firing any other events.

References

[1] M. Bartlett, G. Littlewort, B. Braathen, T. Sejnowski, and J. Movellan. A prototype for automatic recognition of Spontaneous facial actions. In S. Becker, S. Thrun, and K. Obermayer, editors, Advances in Neural Information Processing Systems, volume 15, pages 1271– 1278, 2003.

[2] T. N. Bhaskar, F. T. Keat, S. Ranganath, and Y.V. Venkatesh.Blink detection and eye tracking for eye localization. In Proc. TENCON 2003, volume 2, pages 821– 824, 2003.

[3] Haro, M. Flickner, and I. Essa. Detecting and tracking eyes by using their physiological properties, dynamics, and appearance. In IEEE Internationa Conference on Computer Vision and Pattern Recognition, volume 1, pages 163–168,2000.

[4] Q. Ji, H. Wechsler, A. Duchowski, and M. Flickner. Editorial: special issue: eye detection and tracking.Comput.Vis.Image Underst., 98(1):1–3, 2005.

[5] D. G. Lowe. Distinctive image features from scale-invariant key points. Int. J. of Computer Vision, 60(2):91–110, 2004.

[6] M. Valstar, M. Pantic, Z. Ambadar, and J. Cohn. Spontaneous vs. posed facial behaviour: Automatic analysis of brow actions. In Int. Conf. On Multimodal Interfaces, pages 162–170, 2006.

