

A Practical System for Laser Pointer Interaction on Large Displays

Benjamin A. Ahlborn
*
IDAV, Dept. of Comp. Sci.
University of California, Davis
Davis, CA 95616
baahlborn@ucdavis.edu

David Thompson
Sandia National Labs
Livermore, CA 94550
dcthomp@sandia.gov

Oliver Kreylos
IDAV, Dept. of Comp. Sci.
University of California, Davis
Davis, CA 95616
okreylos@ucdavis.edu

Bernd Hamann
IDAV, Dept. of Comp. Sci.
University of California, Davis
Davis, CA 95616
bhamann@ucdavis.edu

Oliver G. Stadt
IDAV, Dept. of Comp. Sci.
University of California, Davis
Davis, CA 95616
ogstaadt@ucdavis.edu

ABSTRACT

Much work has been done on the development of laser pointers as interaction devices. Typically a camera captures images of a display surface and extracts a laser pointer dot location. This location is processed and used as a cursor position. While the current literature well explains such a system, we feel that some important practical concerns have gone unaddressed. We discuss the design of such a tracking system, focusing on key practical implementation details. In particular we present a robust and efficient dot detection algorithm that allows us to use our system under a variety of lighting conditions, and allows us to reduce the amount of image parsing required to find a laser position by an order of magnitude.

Categories and Subject Descriptors: H.5.2 [Information Interfaces and Presentation]: User Interfaces, Input devices and strategies; I.3.1 [Hardware Architecture]: Input Devices; I.4.1 [Digitization and Image Capture]: Camera Calibration, Imaging Geometry

General Terms: Algorithms, Human Factors

Keywords: Laser Pointer, Tiled Displays, Interaction

1. INTRODUCTION

Current supercomputer simulations of complex physical phenomena are generating massive data sets that continue to grow in size. One approach to harnessing the rich information content hidden in such data sets is the use of large-scale display environments including large tiled display

*Institute for Data Analysis and Visualization

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

VRST'05, November 7–9, 2005, Monterey, California, USA.
Copyright 2005 ACM 1-58113-098-1/05/0011 ...\$5.00.

walls and Virtual Reality caves. Sandia and UC Davis use tiled displays to explore large, complex simulations because they can show fine detail while keeping the entire model in view. Currently, Sandia's input devices are limited to the mouse and keyboard of a single, small console. Typically, one person will sit at the console while others direct this person how to pan, rotate, scale, or otherwise manipulate items displayed. This is obviously unsatisfactory. Nothing on the market meets all of our needs and all security requirements, so we have begun developing our own interaction device. After an initial consideration of the physical techniques available, we decided to use cameras to capture the motion of a laser pointer dot on the display surface. In development of this system we felt that the implementation of an efficient and robust laser pointer detection algorithm was something that is lacking from current literature.

Laser pointers combined with camera arrays for use as interaction devices have become an increasingly active topic in the field of Human Computer Interfaces (HCI) and Computer Vision (CV). A system for controlling slide show presentations using a laser pointer is presented by Sukthankar, Stockton and Mullin [5], including a simple method for generating homographies between cameras and planar display surfaces. A single-pointer system for executing more complicated tasks is presented by Olsen and Nielsen [4]. This paper achieves good results, but is geared at interaction with widget based environments, in particular the XWeb interface. Additionally, it requires the user to be familiar with a series of specific interaction techniques. We desire a system that allows interaction with more general application domains, in particular, navigation and exploration of large data sets. Methods for tracking multiple laser pointers are provided by Oh and Stuerzlinger [3]. While our system does not implement the tracking of multiple laser pointers, this paper contains a good discussion of laser pointers as input devices in general. Each of these discuss systems which use a single camera on a relatively small display. A system that is scalable with respect to display size, resolution, and number of users is presented by Davis and Chen [1]. The setup which the authors present uses a single processing computer for each camera. In a situation requiring a large number of

cameras this is unreasonable. Due to bus bandwidth limitations, the use of multiple computers for pointer detection is unavoidable, however, more than one camera can be utilized on a single machine.

We present an overview of our system with emphasis on describing in detail a robust and efficient method for detecting the laser pointer. Our method uses image subtraction to be able to robustly classify the pixels that compose the laser pointer in an image. We use a predictive tracking method to reduce the amount of image processing that must be done to find the laser pointer.

2. SYSTEM IMPLEMENTATION

Our system uses a group of cameras to detect the position of a laser pointer dot on a display surface. This location is transformed to display coordinates using a planar homography [5]. This point is interpreted as a cursor position and used to control mouse based interaction. We handle the problem of button functionality by using an IR transmitter from a standard remote control and using an IR detector to detect the transmitted button presses. We have developed a set of tools that provide calibration, detection, and interaction using these devices.

2.1 System Setup

The current implementation of our system is running on a 2.66 GHz Pentium IV with 1 GB of memory under Linux kernel 2.4.21. The display system is a 3×2 tiled display. Each tile has a resolution of 1024×768 pixels, providing a total resolution of 3072×1536 pixels. We are using four Canon VC-C4 cameras attached to WinTV frame grabbers. The frame grabber provides 720×480 24-bit color images at 15 fps. The laser pointer is a typical class IIIa red laser pointer. A Linux Infra-Red Controller (LIRC) device is used for detection of button presses. We are currently using front mounted cameras that face the tiled display. In general placing the cameras behind the display is preferable; however, our display has rear baffles which would occlude the cameras.

2.2 Calibration

The calibration of our system occurs in two stages. The first stage requires the determination of radial and tangential distortion coefficients for each camera lens. The second stage involves the generation of a homography between each camera's pixel space coordinates (PSC) and the display normalized device coordinates (NDC).

2.2.1 Lens Distortion Calibration

To correct the lens distortion we use the model described in the OpenCV documentation [2]:

$$\begin{aligned} x_u &= x_d + x_d[k_1 r^2 + k_2 r^4] + [2p_1 x_d y_d + p_2(r^2 + 2x_d^2)] \\ y_u &= y_d + y_d[k_1 r^2 + k_2 r^4] + [2p_2 x_d y_d + p_1(r^2 + 2y_d^2)] \\ r^2 &= x_d^2 + y_d^2 \end{aligned}$$

The radial and tangential distortion coefficients are represented in non-linear equations by k_1, k_2 and p_1, p_2 . These parameters are found by displaying horizontal and vertical lines, one at a time, on the display surface. Each camera captures an image and extracts a set of pixels which compose the line. These lines will appear curved in the camera

image, due to lens distortion. A modified Newton optimization calculates the distortion coefficients for each lens by minimizing the error of a least squares linear fit on each observed line after correcting it using the above equation.

2.2.2 Homography Generation

The relationship of a point, (x, y) in the display NDC and a point (X, Y) in a camera's PSC can be expressed using a 3×3 homography matrix as described by Sukthakar, Stockton and Mullin [5]. This homography is calculated using $N \geq 4$ point correspondences between the display NDC and camera PSC. These point correspondences are determined using the intersections of the horizontal and vertical calibration lines that were captured for the lens distortion coefficient calibration. Since the lines are displayed one at a time there are no issues with determining the orientation of a camera.

2.3 The Interaction Servers

The *Interaction Servers* are applications that generate interaction messages and send them to a client application via network sockets. We have implemented two such servers. The *Button Interaction Server* detects button events using a LIRC device. The *Cursor Interaction Server* uses a set of cameras to detect a laser pointer dot on the display surface. It is in the latter application that we have implemented our detection algorithm. This algorithm consists of a robust solution for detecting the laser pointer and a prediction algorithm that allows us to reduce the overhead of searching for the laser pointer.

2.3.1 Laser Pointer Detection

The detection of the laser dot is an issue that is practically challenging. Our system needs to work in all manner of lighting environments. By adjusting the shutter rate, gain, exposure and various other settings on each camera it is possible to ensure that the laser pointer is both brighter than the surrounding area in the camera image and displays a minimal blurring effect between frames. This; however, is not enough to ensure that the dot can be accurately detected using a constant brightness threshold. The problems observed with using a constant threshold are separated into two groups:

- Spatial intensity variations. With large displays and cameras that must be placed high at awkward angles to the display, the reflectance distribution function of the laser pointer and display wall causes large variations in the measured intensity of the laser pointer across the tiled display. This happens because the angle between the laser beam incidence and the camera's view vector varies significantly as a function of where on the tiled display the laser beam hits. This can cause situations in which the display at certain parts of the camera image is brighter than the laser pointer in other parts of the camera image. A constant threshold value is unable to accurately classify the laser pointer in this case.
- Temporal intensity variations. The captured images may have significant intensity variations from frame to frame, since they are not synchronized to the refresh rate of the projectors. This can render any automatic

gain correction useless. A constant threshold brightness used to identify the laser dot must take these variations into account. Additionally, things such as bright flashing lights in the background can cause significant detection problems. One of the tiled display systems at UC Davis has LED indicators attached to the sonar detection system that surrounds the display. When using a simple threshold value these flashing lights will interfere with the pointer detection.

The solution we have come up with is the use of a background removal process. Prior to starting the system each camera captures a series of images of the display while it is set to show a completely white screen. A *threshold image* for each camera is generated using spatial and temporal data over this series of images. Each pixel in the threshold image is assigned a value that is the maximum intensity of a defined region around that pixel over all images in the sequence. When performing detection the laser pointer is classified by finding pixels that are a set brightness above this threshold image.

The use of the threshold image as reference allows us to handle the described spatial intensity variations. The use of temporal data handles the problems with temporal intensity variation. As long as enough frames are captured to ensure that each pixel has been captured at its brightest point, these situations will not produce artifacts in the detection. The spatial portion of the sampling region is used to prevent problems from slight vibrations of the camera. If the cameras slightly vibrate from outside sources and no spatial domain is used for the threshold image then potentially large amounts of pixels will be misidentified as the laser pointer due to misalignment between the actual camera image and the threshold image. This technique has the advantage of allowing semi-dynamic backgrounds and being flexible in a variety of lighting conditions. It has the downside that it requires generation and storage of an image for each camera and that drastic changes in lighting condition as it runs can throw it off.

The laser pointer detection algorithm is similar to that used by Oh and Stuerzlinger [3]. An image is grabbed from each camera. Pixels are classified as part of the laser pointer using the threshold image. The average location of all identified pixels, weighted by their intensity minus the intensity of the threshold image at that position, is calculated and used as the dot location in the camera image. The pixel space location is then mapped to the NDC of the display. Once the NDC location for a single camera image is determined, it is used as the cursor position. This position is then sent over a network socket and the entire process is repeated.

2.3.2 Acceleration

The image processing stage of the detection cycle can become a bottleneck in systems requiring large number of cameras. Searching the entire camera image to find a laser dot which covers a very small portion of the image can cause unnecessarily high memory and CPU load. Also, it is very likely that the laser dot will be visible by only one of the cameras at a time. Checking camera images in which the dot is not visible consumes a lot of processing time with no added advantage. We implement an acceleration scheme that uses a prediction of the next laser pointer position. By predicting the laser pointer position we are able to determine which cameras are most likely to view the laser pointer and

then search only a small sub-image around the predicted location.

The prediction of the next laser pointer position is based on its previous positions. When ever the laser pointer is found, its position in NDC is recorded. The prediction calculation varies on the number of positions that have been previously recorded¹. Let $[x_0, y_0]$ be the position being predicted. Let $[x_i, y_i]$ be the position captured i frames ago. Then $[x_0, y_0]$ is calculated depending on the number of previous positions as follows:

- 0 positions: brute-force search of all camera images
- 1 position : $[x_0, y_0] = [x_1, y_1]$
- 2 positions: $[x_0, y_0] = 2[x_1, y_1] - [x_2, y_2]$
- 3 positions: $[x_0, y_0] = 2.5[x_1, y_1] - 2[x_2, y_2] + 0.5[x_3, y_3]$

Once the predicted location is calculated it is possible to eliminate some cameras from being searched. This is accomplished by transforming the predicted location into the pixel coordinates of each camera using the display to camera homography. This can be calculated as the inverse of the camera to display homography or at calibration-time directly. If the pixel space position lies outside the camera image then this camera image does not need to be searched.

The next step involves prioritizing which of the remaining cameras images should be searched first. This is accomplished by projecting the center of each of the remaining camera's images onto the display using the camera to display homography. Cameras whose center point is closer to the predicted point are given priority. This calculation is done in NDC because it would not be consistent to use pixel distances with cameras that have different resolutions.

Now that we have a prioritized search list, we can search a sub-image region around the predicted location in each camera's image. If the laser pointer is found in any image, the search is ended and the detection history is updated. If the pointer is not found in any of the sub-image searches, then the previously recorded points are discarded and the system defaults to a brute-force search of all camera images. The priority from the sub-image search is still used in the brute force search. Those images that were initially discarded because the predicted location was outside their image plane are given lowest priority. Again, if the laser pointer is found in any image, the search is ended and the detection history is updated.

2.4 Client Applications

The *Client Applications* are applications which receive interaction messages from the Interaction Servers and translate them into application interaction. We have implemented two client applications that allow use of unmodified applications with our system. One works with the DMX desktop environment. It receives interaction events from the interaction servers and translates them into synthetic X events that allow interaction with the desktop. The other allows use of OpenGL applications via Chromium. In this case the client application receives interaction events from the interaction servers, scales them to the extents of the OpenGL window, and sends synthetic X events to this window.

¹Only the last 3 positions are actually saved

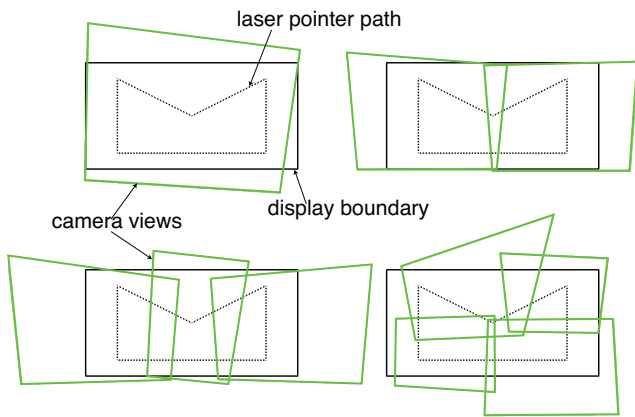


Figure 1: Camera layout for one, two, three, and four casually-aligned cameras.

3. RESULTS

The current implementation of this system uses 4 cameras, each with a resolution of 720×480 . The maximum frame rate we are able to get from the frame grabbers attached to these cameras is 15 [fps]. This total pixel area at this rate is not enough to make image searching a bottleneck and we are able to achieve throughput near the frame rate of the cameras. This, however, is not the case with faster, higher resolution cameras. In order to test the effectiveness of our approach we measured the amount of time spent reading images and the amount of time spent doing the rest of the detection loop. We then evaluate the percentage of that time reading the images vs performing the rest of the loop. A shorter period of time in the loop and more time reading means that the detection was performed quickly, so more time was spent waiting for images from the frame grabber. These measurements were made using the Linux `gettimeofday` system call. We took these measurements using 1, 2, 3 and 4 cameras at a time. The system tracked a simulated laser position that moved over a set path at a constant 75 [pix/s]. The layout of these configurations is illustrated in Figure 1. Three trials were run for both the optimized and brute-force algorithms. The averages of the trials are shown in Table 1. It can be seen that in our implementation the percent of time spent updating the history and searching the camera images is an order of magnitude less than that of the image searching in the brute-force approach. Additionally, as the number of cameras increases, this percent increases far more slowly than it does in the brute-force algorithm. This will allow us to maintain higher frame rates when using faster, higher resolution cameras, where as the brute-force method would be come a limiting factor.

4. CONCLUSIONS AND FUTURE WORK

We have described a system for using cameras and a laser pointer for mouse-based interaction. We have presented in depth details on how to implement a robust and efficient algorithm for detecting the laser pointer. In particular, our algorithm focuses on being able to work in a variety of lighting conditions and with a semi-dynamic background. Additionally, our algorithm presents a method for reducing the

Number Of Cameras	1	2	3	4
Brute-force Read	93.38	90.60	87.80	85.90
Brute-force Detect	6.34	9.28	12.42	14.69
Targeted search Read	99.89	99.88	99.86	99.86
Targeted search Detect	0.09	0.11	0.12	0.13

Table 1: Percent of total time spend reading from cameras and in the rest of the detection cycle for target and brute-force searches. The values are averages from three separate test runs.

overhead involved in searching for the laser pointer by using a predictive searching. These issues have not been the focus of previous work in this area.

In addition to this work, we would like to develop calibration tools that would allow our system to work on non-planar and faceted displays. We would also like to extend our implementation to handle cursor tracking from multiple computers running camera detection. Bandwidth will eventually become the bottleneck in the system and limit the number of cameras that can effectively be used on a single machine. Integration of the ideas presented by Davis and Chen [1] would allow for this.

5. ACKNOWLEDGEMENTS

Authors affiliated with Sandia were supported by the U. S. Department of Energy, Office of Defense Programs. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed-Martin Company, for the United States Department of Energy under contract DE-AC04-94-AL85000. This work is also supported by NSF under contract ACI 9624034 (CAREER Award) We gratefully acknowledge the support of the W. M. Keck Foundation provided to the UC Davis Center for Active Visualization in the Earth Sciences (KeckCAVES), and thank the members of the Visualization and Computer Graphics Research Group at the Institute for Data Analysis and Visualization (IDAV) at UC Davis.

6. REFERENCES

- [1] J. Davis and X. Chen. Lumipoint: Multi-user laser-based interaction on large tiled displays. *Displays*, 23(5), 2002.
- [2] Intel Corporation. Open source computer vision library reference manual, December 2000. <http://sourceforge.net/projects/opencvlibrary/>.
- [3] J. Oh and W. Stuerzlinger. Laser pointers as collaborative pointing devices. In *Proceedings of Graphics Interface 2002*, pages 141–149, 2002. <http://citeseer.ist.psu.edu/oh02laser.html>.
- [4] D. R. Olsen, Jr. and T. Nielsen. Laser pointer interaction. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 17–22. ACM Press, 2001.
- [5] R. Sukthankar, R. Stockton, and M. Mullin. Smarter presentations: Exploiting homography in camera-projector systems. In *Proceedings of the International Conference on Computer Vision*, 2001.