Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Optical Touch Screen with Virtual Force

Hong Zhang
Department of Mechanical Engineering
Rowan University
Glassboro, NJ 08028 USA
zhang@rowan.edu

***Abstract*—This paper presents a novel approach of using stereovision or pseudo-stereovision to generate touch screen experience. A pair of digital cameras or a set of a digital camera and a mirror is mounted at the corners of a computer display. Algorithms are presented to generate the positions of the pointers when they are close to the screen. The paper also proposes to use positions, velocities and accelerations of a pointer to generate virtual forces in the active space of the touch screen. The user interface will be friendlier by applying the rich information obtained from the input methods presented in this paper.**

***Keywords*—optical touch screen, stereovision, pseudo-steresovision, virtual force**

I. INTRODUCTION

The way we interact with computer strongly influences how we use the technology. The traditional input methods such as using a keyboard or a computer mouse are still in dominance. Meanwhile, new methods have been explored and developed to provide more diversified or more intuitive user experiences. These include touch screen, voice recognition and brain wave detection. Among them, touch screen is the most popular approach, especially after the introduction of first the Palm Pilot, then the Windows Tablet PC, and most recently the Apple iPhone.

A traditional touch screen covers the entire displaying surface with a matrix of resistors or capacitors. Special circuits are used to capture the changes of the resistances or capacities of the matrix due to the users' touch. The changes are then converted to cursor positions accordingly. The resistive or capacitive technology requires the use of special materials such as indium tin oxide to be both transparent and conductive. However, the supply of such materials is dwindling fast and the cost is increasing dramatically. Meanwhile, the amount of materials used on such touch screens is almost proportional to their sizes. In compact equipment such as PDAs and cell phones, or in special applications such as public information kiosks, costs of such touch screens can be justified. In regular office or household use, price is often a hurdle that prevents them from being widely adopted.

Different approaches of touch screen technology are developed to overcome the difficulties associated with the resistive or capacitive technologies. For example, an ultrasonic method was proposed by Katsuki et al. to take advantage of surface acoustic waves [5]. Meanwhile, optical based touch screen technology gained renewed interest. It was introduced early but did not gain momentum due to the cost of the digital cameras at the time. More recently, the cost of single chip digital cameras has dropped to the level of a keyboard or a computer mouse. It is feasible to take cameras as the building blocks of today's input technology.

Several algorithms have been introduced within the optical input technology. Han proposed a low-cost multi-touch sensing system using frustrated total internal reflection [3]. Wilson used two cameras to transform an acrylic plastic to a touch screen [7]. However, these two methods require the cameras to be placed on the opposite side of the screen from the user. They inevitably increased the thickness of the display and are not suitable for everyday household or office use.

Another popular approach is to put the camera at the user's side. For example, Cheng and Takatsuka used a mono-view camera from the user's side to track a laser pointer or a fingertip [1]. Meanwhile, a popular do-it-yourself project proposed by Lee [4] is to use the infrared camera of a WII remote to track the motion of an infrared pointer. Nevertheless, the view of the user side cameras can be easily blocked by the body of the users and needs calibrations whenever the display is moved. Except the large projector screens, these methods are hard to be adopted on regular desktops or laptops.

Meanwhile, stereovision has long been used in robot navigation to control manipulation or maneuver [2]. Koh, Won and Bae proposed a virtual touch screen using stereovision and see-through head mounted display [6]. However, the set up is not suitable for daily application and the image processing is complicate due to the unstructured and noisy background.

In this paper, we propose to use stereo- or mono-vision from the corners of the displays. As we will explain later, this approach is simple to implement, inexpensive to equip, and not sensitive to the wear and tear of the screen. Comparing to existing touch screen technologies, it is also easy to scale up or down for different sizes or height/width ratios. Further, we can even generate virtual forces in the active input space and provide more vivid and intuitive user experiences.

The structure of this paper is as follows. After this introduction, we will provide two approaches of optical touch screens, one based on stereovision and the other one on pseudo-stereovision. Then we will introduce the idea of virtual

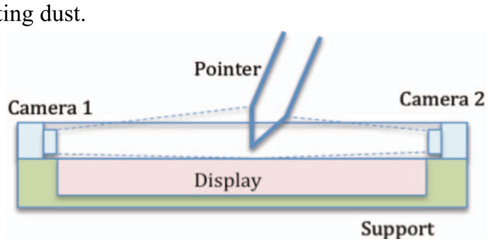
978-1-4244-2794-9/09/$25.00 ©2009 IEEE

force to be used in touch screen input. After that, a quick discussion will be followed in the final section.

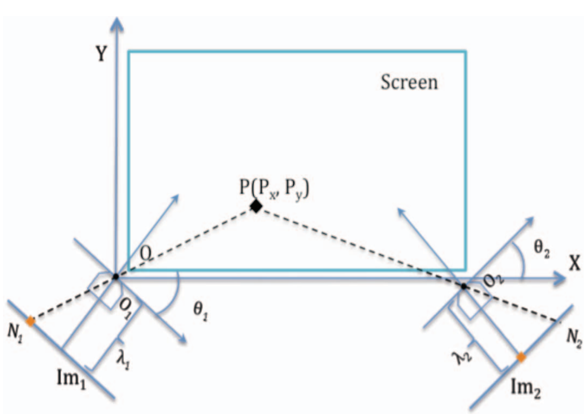
II. Pointer Locator Using Stereovision

A. System Set Up

In Pointer Locator Using Stereovision system, two digital cameras are used. Each camera comes with a viewing angle that is equal to or greater than 90 degrees. As shown in Figure 1, a supporting structure encloses the flat screen display. At two adjacent corners, two cameras are mounted just above the surface of the display. They are positioned toward the center of the display such that the overlapped viewing field covers the entire screen surface. For convenience of illustration, we assume the cameras are located at the lower left and lower right corners as depicted in Figure 2. In applications, the cameras are generally put on the top two corners to prevent occlusions by hand or to keep the camera lenses from collecting dust.



[illegible]



[illegible]

For convenience of analysis we first build a coordinate system. As seen in Figure 2, the origin of the system is at the focal point of the lower left camera. Its X and Y axes are parallel to the horizontal and vertical edges of the screen respectively. At this section, we only analyze the pointer positions. That is, we will only study the planar (X-Y) movement of the pointer projected at the surface of the display. Therefore, we assume a linear camera and ignore the Z-coordinate. The focal point of the camera is offset by $-s_{1x}$ and $-s_{1y}$ at horizontal and vertical directions respectively from the lower left corner of the screen. Then the position of the corner is $\mathbf{s}_1=(s_{1x}, s_{1y})$. The lower right corner is $\mathbf{s}_2=(s_{2x}, s_{2y})=(L_H+s_{1x}, s_{1y})$ where L_H is the width of the screen. Likewise, the upper right corner is $\mathbf{s}_3=(s_{3x}, s_{3y})=(L_H+s_{1x}, L_V+s_{1y})$ where L_V is the height of the screen.

B. Pointer Location Using Stereovision

Assuming the optical axis of the left camera is rotated from X-axis by angle θ_1, we can obtain the transformation matrix of the camera from screen-based coordinate system to camera-based coordinate system as

$$A=\begin{bmatrix}\cos\theta_1 & -\sin\theta_1\\ \sin\theta_1 & \cos\theta_1\end{bmatrix}.$$

Likewise, assuming the focal point of the right camera is located at $\mathbf{d}_2=(d_{2x}, d_{2y})$ with angle θ_2, the transformation matrix is

$$A=\begin{bmatrix}\cos\theta_2 & -\sin\theta_2 & d_{2x}\\ \sin\theta_2 & \cos\theta_2 & d_{2x}\\ 0 & 0 & 1\end{bmatrix}.$$

Therefore any point $P=(P_x, P_y)$ in the screen can be transformed to

$$P_1=\begin{bmatrix}P_{1x}\\ P_{1y}\end{bmatrix}=A_1P=\begin{bmatrix}P_x\cos\theta_1-P_y\sin\theta_1\\ P_x\sin\theta_1+P_y\cos\theta_1\end{bmatrix}$$

at camera 1 and

$$P_1=\begin{bmatrix}P_{2x}\\ P_{2y}\end{bmatrix}=A_2P=\begin{bmatrix}P_x\cos\theta_2-P_y\sin\theta_2\\ P_x\sin\theta_2+P_y\cos\theta_2\end{bmatrix}$$

at camera 2. Please note that we implied the change of P from (P_x, P_y) to $(P_x, P_y, 1)^T$ in calculation and the change back in expression whenever necessary. We assume a pinpoint camera model with focal length θ_0. For camera 1, the projection of the point P on the image plane can be easily measured from the image as $N_1=n_1*c_{px1}$ where n_1 is the pixel position at the image plane and c_{px1} is the size of each pixel for camera 1. Hence, we can easily obtain the following relationship using similar triangles:

$$\frac{P_x\sin\theta_1+P_y\cos\theta_1}{P_x\cos\theta_1-P_y\sin\theta_1}=\frac{-\lambda_1}{N_1}$$

Likewise, we can find similar result for camera 2 as:

$$\frac{P_x\sin\theta_2+P_y\cos\theta_2+d_{2y}}{P_x\cos\theta_2-P_y\sin\theta_2+d_{2x}}=\frac{-\lambda_2}{N_2}$$

By solving Equations 1 and 2 we can get the original position of $P(P_x, P_y)$ as

$$\begin{cases}P_x=\dfrac{C_1B_2-C_2B_1}{A_1B_2-A_2B_1}\\ P_y=\dfrac{A_1C_2-A_2C_1}{A_1B_2-A_2B_1}\end{cases}$$

where

$$A_1=N_1\sin\theta_1+\lambda_1\cos\theta_1,\quad A_2=N_2\sin\theta_2+\lambda_2\cos\theta_2,$$
$$B_1=N_1\cos\theta_1-\lambda_1\sin\theta_1,\quad B_2=N_2\cos\theta_2-\lambda_2\sin\theta_2,$$
$$C_1=0,\quad C_2=-\lambda_2d_{2x}-N_2d_{2y}$$

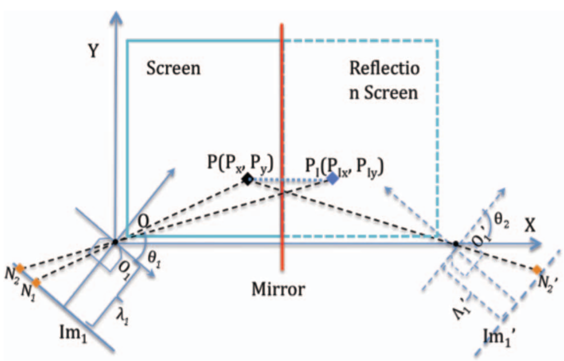
The corresponding cursor position of the pointer is obtained as $P_s=(P_{sx}, P_{sy})=((P_x-s_{1x})/s_{px}, (P_y-s_{1y})/s_{py})$ where s_{px} and s_{py} are pixel sizes of the display unit at both X and Y directions. Please note that the values of s_{1x}, s_{1y}, d_{2x}, d_{2y} can be obtained from initial calibration. However, they will not change even if the display is moved since the cameras and the screen are both fixed to the same supporting structure.

III. Pointer Locator Using Monovision

A. *Pseudo-Stereovision*

In normal stereovision, the environment surrounding the interested features is often full of image noises. However, the scenes observed by the pointer detecting cameras are simple and constructed. Therefore, we can use one camera instead of two to achieve the same result.

As seen in Figure 3 the camera on the right is removed. Instead, a mirror is put on the edge of the screen. For the convenience of illustration, we cut the width of the screen to half of that is shown in Figure 2. The mirror is configured to be parallel to the left edge of the screen, or the Y-axis of the screen-based coordinate system. Therefore we can have a mirror image of the camera (virtual camera) with the exact same physical characters as the real one. By combining the physical camera and its reflection (the virtual camera) together, we can obtain a pseudo-stereovision.



[illegible]

From the set up shown in Figure 3, it is obvious that

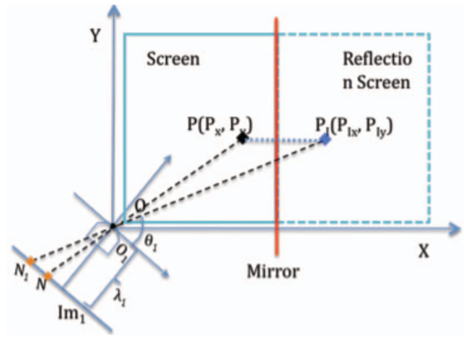
$$\begin{cases} \theta_2 = -\theta_1 \\ \lambda_2 = \lambda_1 \end{cases} \quad [illegible]$$

The focal point or origin of the virtual camera frame is at $(d_{2x}, d_{2y})=(2L_H+2s_{1x}, 0)$. Meanwhile, as seen from Figure 3, the mirror image of the pointer $P(P_x, P_y)$ is $P_1(P_{1x}, P_{1y})$. Its projection to the image plane of the physical camera is N_2. As depicted from Figure 3, the mirror image of the projection of P1 on the physical camera is the projection of the point P on the virtual camera. It is clear that $N'_2=-N_2$.

If we substitute the above values to Equation 5, we can easily find the position of pointer $P(P_x, P_y)$ and its corresponding cursor position $P_s=(P_{sx}, P_{sy})$.

B. *Monovision With Mirror Image*

Alternatively, we can use a different approach. That is, instead of using the virtual camera and convert back to the case of stereovision, we can process the mirror image of the pointer at the physical camera directly.



[illegible]

First, using the same set up of the coordinate system, the position of $P_1(P_{1x}, P_{1y})$ can be found as $(2L-P_x, P_y)$ where L is the horizontal distance from the origin O to the mirror. Then we can obtain the equations similar to Equations 3 and 4:

$$\frac{P_x \sin\theta + P_y \cos\theta}{P_x \cos\theta - P_y \sin\theta} = \frac{-\lambda}{N_1} \quad [illegible]$$

$$\frac{(2L-P_x)\sin\theta + P_y \cos\theta}{(2L-P_x)\cos\theta - P_y \sin\theta} = \frac{-\lambda}{N_2} \quad [illegible]$$

where λ is the focal length of the physical camera, λ is the angle of the camera, and N_1 and N_2 are the projected positions of P and P_1 on the image plane.

Solve for P_x and P_y, we have

$$\begin{cases} P_x = \dfrac{(\lambda^2 - N_1N_2)\sin 2\theta + 2N_2\lambda\sin^2\theta - 2N_1\lambda\cos^2\theta}{(\lambda^2 - N_1N_2)\sin 2\theta - (N_1+N_2)\lambda\cos 2\theta} L \\ P_y = \dfrac{N_1\sin\theta + \lambda\cos\theta}{\lambda\sin\theta - N_1\cos\theta} P_x \end{cases} \quad [illegible]$$

Note that the result is simpler compare to the true stereovision method due to the equality of focal length λ and rotation angle [illegible]. Meanwhile, we need to cover the combined surface of both the physical screen and the reflected one, so we can calibrate the angle λ to be λ/4. In this case, the result can be further simplified as

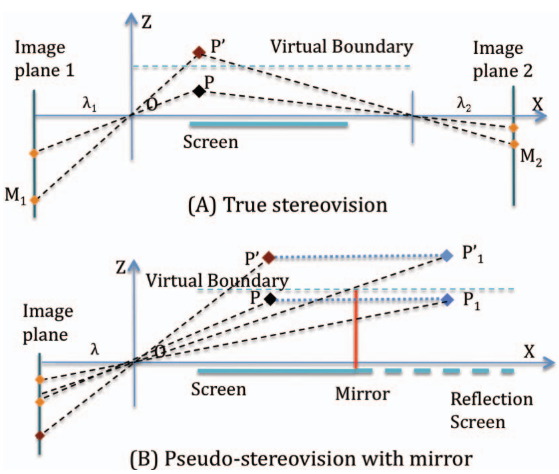
$$\begin{cases} P_x = \dfrac{(\lambda - N_1)(\lambda + N_2)}{\lambda^2 - N_1N_2} L \\ P_y = \dfrac{\lambda + N_1}{\lambda - N_1} P_x \end{cases} \quad [illegible]$$

C. *Virtual Boundary*

Besides the simplicity of calculating pointer position, another advantage of the pseudo-stereo method is its ability to quickly filter unwanted motions. Sometime, the pointer is

away from the vicinity of the screen but is still close enough to be picked up by the cameras. This can easily happen when a user is pondering for the next action or is talking in front of the screen. As shown in Figure 5A, in true stereovision, a virtual boundary can be set up in the algorithm to filter out the unwanted movements. Any point P within the virtual boundary is considered legitimate and is processed accordingly. Its motion will be reflected as the movement of the cursor on the screen. If the pointer is outside the virtual boundary, like the position P' in Figure 5A, it can be considered idle and an update is not necessary. However, in the background, the video stream is constantly processed to extract the positions of P at both cameras. The distance of the pointer from the screen surface needed to be constantly calculated and compared with the virtual boundary.

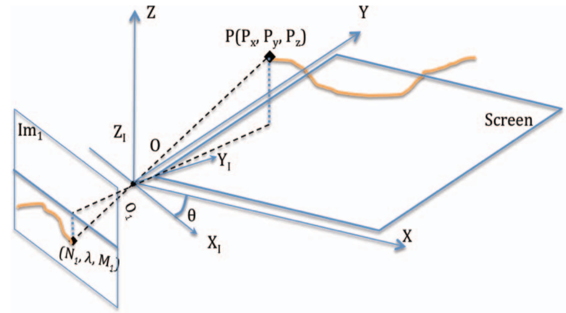
On the other hand, with the method of monovision using mirror, we can set up a virtual boundary by combining the height of the mirror and the viewing angle of the camera. As shown in Figure 5B, when the pointer is above the virtual boundary, the image of the pointer P'_1 will not be picked up by the camera even the pointer P' itself is within the range. That is, only one image feature will be obtained instead of two. A quick on-off logic can be easily used to dismiss this single point without calculating the position and distance first. Since the duration of the pointer in use is often far less than when it is not, demand on computing power will be greatly reduced.



IV. Action Space and Virtual Force

In previous methods, only the X-direction projection (parallel to the screen) on the camera image plane is used. It only provides information on the position of the pointer. Move one step forward, we can take advantage of the Z-direction projection (perpendicular to the screen) of the image and the time derivatives of the projections for more advanced features. One such application is to set up an action space immediately outside the touch screen. In the action space, we can construct virtual forces and use them to achieve the results such as push, pull, stroke, or other similar motions.

First, let us expand the aforementioned model to a 3-dimensional space with the Z-axis perpendicular to the display screen and pointing away from it. This 3D action space is enclosed by the display screen on bottom and the virtual boundary on top. As seen in Figure 6, the pointer can freely move in this space. The projection of the pointer gives a planar trajectory on the camera image plane.



Then the z position P_z can be easily obtained as

$$P_z = \frac{M}{\lambda} P_x$$

where M is the vertical position of the pointer projected on the image plane.

With the 3D image features obtained, we can easily construct a virtual force function at both directions. A virtual stroke force can be obtained as

$$F_s = K_s \sqrt{a_x^2 + a_y^2}$$

and a virtual push or pull can be considered as

$$F_p = K_p a_z$$

where K_s and K_p are to be obtained from experiment and to be further adapted using self learning algorithms, a_x and a_y are accelerations of pointer at x and y directions, while a_z is its acceleration at the z direction. The variables a_x, a_y and a_z can all be calculated through second order finite differentiation from P_x, P_y, and P_z.

We should point out that a virtual force can be considered as a function of any subset of the position variables and their derivatives. For example, a simple position based force feedback can be obtained as

$$F = K_p P_z^{-1}$$

while a more sophiscated one may look like

$$F = \sum_{i=1}^{3} K_{pi} P_i^{-ai} + K_{vi} V_i^{-bi}$$

where K_{pi}, P_i, a_i, K_{vi}, V_i, and b_i should be determined by the specific program needs.

Comparing with the force sensing of traditional touch screens, the virtual force can give a much greater force range. The depth of the action space can be easily adjusted by the virtual boundary. By using cameras with wide viewing angles, the action space can be large enough to yield many interesting

3D motions, which can be important for active computer games or on screen calligraphy or painting. Meanwhile, the pull or similar motion is absent in many applications. If used, they can greatly improve the user interface.

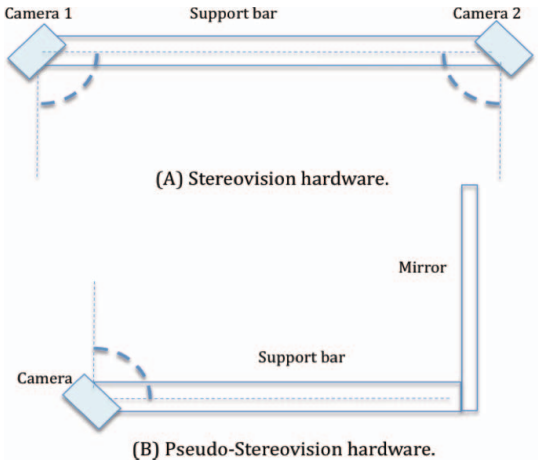
Further, the users do not even need to touch the screen. As long as the pointer or finger is in the action space, its movement will be monitored and reflected on the movement of the corresponding cursor or object in the program.

I. Discussion

In this paper, we introduced a novel method of using stereovision and pseudo-stereovision to locate the position of a pointer on a display unit.

A. Implementaiton

The setup of the system is simple and quick. In the stereovision case, a bar with two cameras is attached to the top of the display screen, as shown in Figure 7A. This bar-camera set can be pre-manufactured for standard screen sizes or tailored for specific purposes. For the pseudo stereovision, an L shaped hardware configuration is used as shown in Figure 7B. For both cases, the viewing angle of the cameras should be equal to or greater than 90 degrees. Both hardware can be installed on any kind of display, such as a desktop monitor, a laptop computer, the screen of an overhead projector, or even a whiteboard. It can be used as either a pre-installation or an aftermarket modification.



2222222222 2222 2222222222

B. Benefit

There are many advantages of the algorithms proposed in this paper.

1. Low cost. It is relatively easy and inexpensive to implement. Thanks to the ubiquitous camera phones and webcams, low-cost high-resolution single-chip digital cameras are widely available today.
2. Durable. There is no physical contact between the camera and the pointer. Therefore, the functionality of the screen will not deteriorate due to wear and tear. Even when the screen is scratched or broken, the touch function will still work.
3. Easy to scale up. The complexity and cost of existing touch screen increase quickly when the screen becomes larger. For example, when the size of the screen doubles, its viewable area quadruples, so is the material used and the computing power needed. In contrast, in the methods introduced in this paper, the same two cameras or camera and mirror combination can be easily scaled up or down to screens with different sizes and height-width ratios.
4. 3D input. As mentioned earlier in this paper, the action space and virtual force can be used to achieve the effect of 3D input and to obtain virtual force feedback.
5. Multi-touch. Existing computer software can easily detect and follow multiple image features. It will be easy to include multi-touch functions. Combined with virtual force, the multi-touch function can make inputting more versatile and powerful.
6. Compact and light weight. Comparing to the existing optical touch screen using behind screen or user side camera, this method is compact in size and can save precious space. The cameras are generally small and will not increase the weight or size of the computer display like many existing touch screens.

C. Future Work

Currently we are working on computer programming to implement the touch screen algorithm proposed in this paper. Experiments will be conducted to test the sensitivity of the cameras under noisy background and to calibrate the coefficients for realistic virtual forces.

References

[1] K. Cheng and M. Takatsuka, "Estimating Virtual Touchscreen for Fingertip Interaction with Large Displays" Sydney, Australia : s.n., 2006. 20th conference of the computer-human interaction special interest group (CHISIG) of Australia on Computer-human interaction: design: activities, artefacts and environments (OZCHI'06). pp. 397-400.

[2] G. Hager, “Calibration-free visual control using projective invariance". 1995. In Proc. of 5th ICCV. pp. 1009-1015.

[3] J. Han, "Low-cost multi-touch sensing through frustrated total internal reflection". 2005 . Proceedings of the 18th annual ACM symposium on User interface software and technology. pp. 115 - 118 .

[4] J. Lee, "Low-Cost Multi-point Interactive Whiteboards Using the Wiimote". [Online] [Cited: 03 01, 2009.], johnnylee.net/projects/wii/.

[5] T. Katsuki, F. Nakazawa, S. Sano, Y. Takahashi, Y. Satoh,. "A Compact and High Optical Transmission SAW Touch Screen With ZnO Thin-Film Piezoelectric Transducers". Oct. 5-8, 2003. 2003 IEEE Symposium on Ultrasonics. Vol. 1, pp. 821-824.

[6] E. Koh, J. Won, and C. Bae. "Vision-Based Virtual Touch Screen Interface". Jan. 9-13, 2008. International Conference on Consumer Electronics. pp. 1-2. ISBN: 978-1-4244-1458-1.

[7] A. Wilson, "TouchLight: An Imaging Touch Screen and Display For Gesture-Based Interaction". State College, PA, USA : s.n., 2004. 6th International Conference On Multimodal Interfaces. pp. 69-76.