Vision, Video, and Graphics (2003), pp. 1–6
P. Hall, P. Willis (Editors)

A novel form of a pointing device

H. Cantzler and C. Hoile

Intelligent Systems Lab, BTexact Technologies, Ipswich, UK

Abstract

This paper presents a novel approach for man machine interaction applying real time computer vision techniques. We use a handheld camera to control the mouse cursor on a computer display. The camera captures an image of the display in its field of view and this can be used to judge the camera's position and orientation relative to the display. The problem is modelled as a plane-to-plane projection (homography). Once the mapping of the display in the camera view to the real world display is known, the intersection between the central axis of the camera and the surface of the display can be computed. The mouse pointer is then moved to the corresponding display position. This calculation can be iterated continuously to update the mouse cursor position as the camera position and orientation changes. The camera can then be used to control the mouse cursor just like a laser pointer controls a laser dot. A prototype has been developed to demonstrate the approach.

1. Introduction

A pointing device like a mouse enables users to control a pointer and interact with a graphical user interface. Some devices are more intuitive and flexible to use than others. Controlling the spot produced by a laser pointer is intuitive and requires no technical skills or previous experience. By contrast, touchpads, trackballs and mice control the mouse pointer only indirectly and also require specific conditions (*e.g.* a mouse requires a supporting surface of a specific texture).

We present a novel pointing device which is very intuitive to use and requires only low-cost simple hardware. Our pointing device employs an off the shelf web camera to control the mouse pointer on the screen. It does this by finding the screen in the camera view. We calculate where the centre of the camera view is on the screen and move the mouse pointer accordingly (see figure 1). Using the device, the mouse pointer can be controlled in the same way as the spot of a laser pointer, by physically orienting the pointing device at the desired location on the screen. Since our control strategy only requires the visibility of a screen display, it can potentially be used to control a pointer on any display device (white paper, see http://www.cefn.com/papers/wand.pdf).

The device is designed to replace conventional 2D pointer devices such as touchpads, trackballs and mice. It can be used just by orienting it freely in 3D space in any environment where a mouse pointer must be controlled on a screen. It would be particularly useful in deskless environments, walk and talk environments, interactive TV, or as an intuitive pointer for next generation interfaces (*e.g.* in video games) and in general as an accessibility tool.

Early research in Human-Computer Interaction (HCI) enabled users to access the computer by moving their eyes [5]. Five electrodes were placed around the eye. The electrodes sensed the eyes movement and transmitted it to the computer. Current computer vision research is mainly concentrated on visual hand/face gesture recognition and tracking systems using video cameras. Gestures such as hand, finger [2, 12], head [1], face [4] or eye [13] movement can be used to control the computer. Even the movement of a mouse like object on a mouse pad can be used [3]. The observed movement is translated into movement of the mouse pointer. The sensor typically consists of a video camera which follows the movement. A commercial application also exists from NaturalPoint. A infrared sensor is mounted on top of the display. It emits infrared light that is reflected by a special marker attached to the users head or finger. So, head or finger movement can be recognised and translated into mouse movement. The mouse pointer is controlled according to the horizontal and vertical movement of the marker relative to the sensor's field of view, which can be quite counter-intuitive. The position of the finger in front of the centre of the dis-

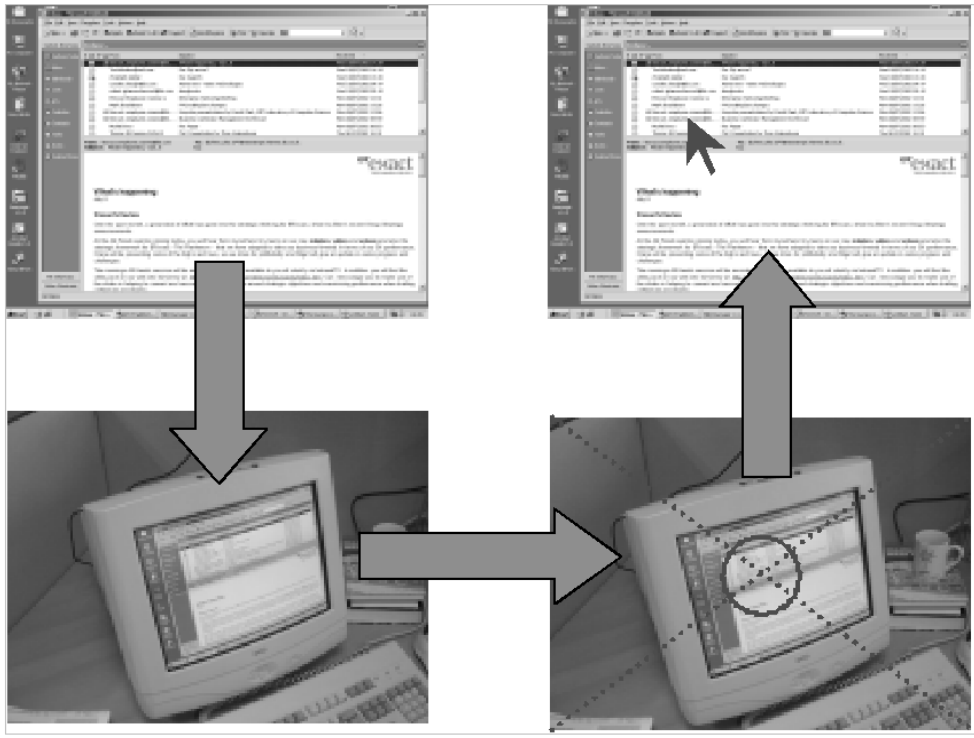

Figure 1: *We identify the screen (top left) in the web-cam image (bottom left), calculate where the centre of the web-cam points at on the screen (bottom right) and set the mouse pointer on the screen accordingly (top right).*

play does not necessary correspond with the centre position of the mouse pointer on the display. The position must be calibrated first. Similarly, the speed of the mouse movement relative to the finger movement must be set. More advanced video based optical tracking systems, which are used for example for real-time gesture recognition [10], overcome those shortcomings.

Our approach avoids these problems since the camera can be oriented at any point on the display in an intuitive and unambiguous way, with no calibration. We use a single handheld camera which points towards the display. Research in robotics has shown us that we can estimate the localisation of a single camera in real-time [9]. However, estimation of the camera pose in real-time is still a challenging task. Our approach uses projective geometry to estimate the position the camera points at. It maps the display in the image to the display in the world. This is a plane to plane transformation and is called homography [6]. To calculate the homography we have to extract 4 corresponding pairs of points from the camera image and the display image.

The paper is organised as follows: In section 2 we present a brief overview about homography. In section 3 we discuss image segmentation and extraction of corresponding regions and points. In section 4 we give details to our prototype implementation and its performance. Section 5 includes the conclusion and section 6 discusses future extensions to our approach.

2. Homography

We model the problem of estimating the mouse pointer position as the estimation of the homography of the screen in the image to the screen in the world. Points p in one plane are mapped into the corresponding points w in the other plane by using a 3 x 3 projection matrix h representing the homography transformation. (see figure 2).

$$\begin{pmatrix} p_x \\ p_y \\ 1 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} w_x \\ w_y \\ 1 \end{pmatrix} \quad (1)$$

Once the homography matrix h is known (or has been computed), any image point can be mapped into the corresponding location on the display's planar surface. The reverse is also true.

The homography matrix h is computed from a set of non-collinear corresponding points. We need to specify 4 point correspondences to obtain an exact solution. However, if

more than 4 such pairs are extracted, then the correspondences might not be fully compatible. Therefore, we have to select the best 4 pairs of points. Each of the 4 image to world point correspondences provides then two equations which are linear in the elements of the matrix h. They are:

$$p_x = \frac{h_{11}w_x + h_{12}w_y + h_{13}}{h_{31}w_x + h_{32}w_y + h_{33}} \quad (2)$$

$$p_y = \frac{h_{21}w_x + h_{22}w_y + h_{23}}{h_{31}w_x + h_{32}w_y + h_{33}} \quad (3)$$

and multiplied out:

$$h_{31}p_xw_x + h_{32}p_xw_y + h_{33}p_x = h_{11}w_x + h_{12}w_y + h_{13} \quad (4)$$

$$h_{31}p_yw_x + h_{32}p_yw_y + h_{33}p_y = h_{21}w_x + h_{22}w_y + h_{23} \quad (5)$$

Figure 2: *The planar display screen in the world (right) is projected onto the planar screen in the image plane (left)*

For 4 correspondences we obtain a inhomogeneous set of 8 equations in 8 unknown. We assume h_{33} equals 1. Imposing the condition is justified since the solution is determined only up to scale. A linear solution for h is then obtained by solving the set of linear equations.

$$\begin{pmatrix} w_{1_x} & w_{1_y} & 1 & 0 & 0 & 0 & -p_{1_x}w_{1_x} & -p_{1_x}w_{1_y} \\ 0 & 0 & 0 & w_{1_x} & w_{1_y} & 1 & -p_{1_y}w_{1_x} & -p_{1_y}w_{1_y} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{4_x} & w_{4_y} & 1 & 0 & 0 & 0 & -p_{4_x}w_{4_x} & -p_{4_x}w_{4_y} \\ 0 & 0 & 0 & w_{4_x} & w_{4_y} & 1 & -p_{4_y}w_{4_x} & -p_{4_y}w_{4_y} \end{pmatrix} \begin{pmatrix} h_{11} \\ h_{12} \\ \vdots \\ h_{31} \\ h_{32} \end{pmatrix} = \begin{pmatrix} p_{1_x} \\ p_{1_y} \\ \vdots \\ p_{4_x} \\ p_{4_y} \end{pmatrix} \quad (6)$$

3. Corresponding points

To estimate the homography we need to extract 4 corresponding points. Extracting corresponding points is a fundamental problem in computer vision. Correspondence techniques can roughly be divided into feature based and correlation based techniques.

Typically, feature-based techniques identify corresponding strong landmarks like corners or line intersections in the images. These landmarks are invariant to projective transformations and lead to precise matches. However, extracting them is computationally rather expensive. Initial edge extraction (*e.g.* Sobel and Canny) is fairly quick. However, the following corner extraction (*e.g.* Harris) or line extraction (*e.g.* Hough) does not deliver results with an adequate speed.

Classical correlation based techniques have often been used for recovering correspondence in images with small transformations. This technique is extreme easy to use. The image is divided in small windows and for each window an characteristic value depending on the intensity values in the window is calculated. Windows in the two images with similar values are matched consecutively. Beside its simplistic use, this technique is also computationally very effective. However, this approach falls short in terms of invariance to projective distortions (one needs to adjust the window size and shape) and colour distortions (camera distortions and motion blur).

For our first prototype we employed an approach which combines the efficient nature of the correlation based technique and the precise invariant properties of the feature based technique.

The first step is to sub sample the data to make the feature extraction in the following steps less computationally demanding, since we need to analyse less data. The sub sampled data is then used to find connected regions with the same colour. We allow small graduated colour changes in the regions to compensate for small illumination changes resulting from inhomogeneous lighting. We exploit the fact that screenshots of office programs or presentation slides consist of strong structural elements such as strong edges and coloured regions. Thus, we can segment the display content and the camera image. Regions are represented by their centre of gravity, size and colour. We select the biggest region b from the computer display device. From the camera image we extract the n biggest regions r_n, since we are not interested in small regions. See the top right picture of figure 3 for an example segmentation.

The next step is to identify the region in the camera image which corresponds with the selected region b on the display (shown in the bottom left of figure 3). We generate a candidate table including all the extracted n regions from the camera image. For each region we then calculate a score $score_i$. The score includes how different the colour of the region r_i is compared to b, how much the region is in the centre of the view and its size. We compare the colours in HSV (Hue, Saturation and Value) space taking H and S into account. The HSV space separates the intensity (or luminance) from the colour information. Our segmentation is therefore robust to changes in illumination. The last two terms exploit the properties that the region in search must be close to the centre of the view, (the camera's central axis must intersect with the screen to position the pointer at all), and that it is unlikely to be very small, (since it must present a viewable display for the human user). Regions on the exterior of the camera image and small regions thus get a penalty. All three terms are weighted with a_1, a_2 and a_3.

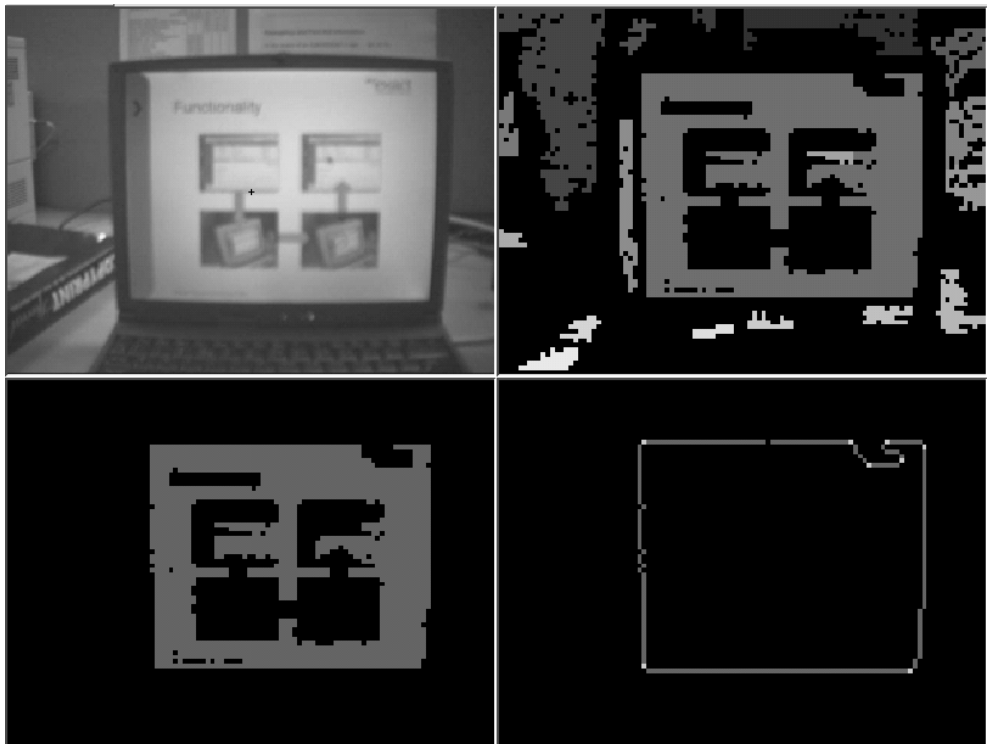

Figure 3: *The top left image shows a computer screen in an office environment with a typical slide of a slide presentation. The image of the scene is segmented into coloured regions on the top right. The biggest region on the display is the white background region. The corresponding region is selected from the extracted regions in the camera image on the bottom left. The bottom right shows the extracted region border and the extracted corner points. Some surplus corner points are extracted, because of an imperfect segmentation. Only the 4 outer corner points are used for estimating the projection, since they are furthest away from each other.*

$$score_i = a_1 colour(b, r_i) + a_2 centre(r_i) + a_3 size(r_i) \quad (7)$$

The image of the scene is segmented into coloured regions on the top right. The region which the highest score is selected as the corresponding region.

At this point, we have identified the region in the camera view which corresponds with the region b on the display. However, to compute the homography we need to identify 4 corresponding points. We identify corresponding points on the two region borders. Firstly, the region border in the camera image is smoothed with a morphological filter. The outline of the border is then tracked and strong corner points (*i.e.* points of high curvature) are extracted. On a display in a working environment there are typically strong structural elements such as corner points. The bottom right image of figure 3 shows an example extraction of the region border with the extraction of corner points. We do the corner point extraction for the regions in the camera image and the display image.

To estimate the homography, we need to restrict the number of the corresponding corner points to 4. We use a simple heuristic here to remove surplus and wrongly extracted corner points that result from small segmentation errors. We select the 4 corresponding corner points which are furthest away from each other. Since we work with corresponding regions, they have the same shape and the corner points that are furthest away from each other, should be the same. Futhermore, using corresponding points that are far away from each other make the estimation of the homography more stable.

In a final step, the positions of the corner points are refined by extracting them from the original data. This compensates for the imprecision introduced by the sub sampling earlier on. Now, that we have a set of 4 corresponding point pairs, we can use them to compute the homography and project the centre of the camera image back onto the plane of the display screen, giving us the correct position for the mouse pointer.

4. Implementation and results

We implemented a prototype which employs the above described techniques in C. The program runs under the Linux operating system.

For image acquisition, we used a low-cost off the shelf Web-Cam from Phillips (PCVC740K "ToUCam Pro"). This is a fairly small camera which can be easily held by a user. It is connected to the computer via a USB 1.1 interface (11MBit/sec). The Linux hardware driver utilise compression to achieve a higher framerates with the webcam. The driver delivers a RGB image.

The Web-Cam delivers images in a resolution of 320 by 240 which is sub sampled by a factor of 3 to 106 by 80 by the prototype. The prototype was tested on a Pentium II based Laptop (400 Mhz). The slide presentation was displayed on the 14.1 TFT Laptop display with the resolution of 1024 by 768. Figure 3 shows a sample slide of the presentation on the top left.

In our prototype experimentation we used presentation slides which had homogeneous regions which were already known. This allows the CPU to be dedicated to the automatic analysis of the camera image, hence maximising the frequency at which the mouse cursor position is updated.

We achieve real-time performance. The update frequency for the mouse pointer position is roughly 10 times per second. The precision of the mouse pointer position estimation is 2~3 pixels, which results in small jumps of the mouse pointer on the screen when the camera is held still. The used region extraction can coupe with moderate illumination changes in the environment.

The prototype works in a range of 40cm until 3m distance to the screen. The lower bound of this range arises because the prototype implementation requires the whole corresponding region, including its corners, to be visible by the camera. At distances over 3m, the display content in the camera image becomes too small and it becomes difficult to identify the region corresponding to the display in the camera image.

5. Conclusions

We have presented the development of a prototype which demonstrates a novel approach for a computer pointing device. The development can have many applications in Human-Computer Interaction (HCI). A camera held by the user is used to control the mouse pointer by moving the camera freely in space in a similar fashion to the way a laser pointer controls a laser dot.

The program locates the computer display in the camera image by finding matching regions following region extraction. Corresponding points on the region border are extracted which are points of high curvature. The homography is estimated with 4 such corresponding pairs of points. Thus, the centre of the camera image can be projected onto the display screen. The mouse pointer is moved to the estimated position, and hence the mouse pointer is controlled. Crucially, this technique is computationally efficient, and could therefore allow the user to use normal office applications whilst achieving a suitable update frequency for the mouse pointer position.

The experimental platform has been designed to accommodate future experimentation with alternative techniques. It is hoped to acquire further investment, and identify appropriate academic or commercial partners to develop a full fledged product based on the ideas presented in this paper.

6. Future development

The technique presented in this paper works by computing the homography with 4 extracted corresponding points in the two views - the camera image and the display image. The chosen method exploits the fact that there is much structured content (large regions). Further work is necessary to identify corresponding points within less structured data such as photographic images and videos. Furthermore, using more than 4 corresponding point pairs would make the results more stable.

Tracking techniques can be used which take advantage of the computations completed in the last iteration of the algorithm to improve the efficiency of mouse pointer position estimation in the next iteration. A fast motion model [8] can be used to track the screen in the camera view in real time. Also, the effects of noise on the mouse pointer position estimation could be reduced using techniques such as Kalman filtering [7, 11] and the effects of small involuntary hand-motions could be smoothed by averaging over several iterations.

References

1. M. Black, F. Berard, A. Jepson, W. Newman, E. Saund, G. Socher, and M. Taylor. The digital office: Overview. *American Association for Artificial Intelligence Spring Symposium on Intelligent Environments, Stanford, USA*, 1998.

2. J. Crowley, F. Berard, and J. Coutaz. Finger tracking as an input device for augmented reality. *International Workshop on Gesture and Face Recognition, Zurich*, 1995.

3. I. Erdem, M. Erdem, Y. Yardimci, V. Atalay, and A. Cetin. Computer vision based mouse. *International Conference on Acoustics Speech and Signal Proceedding, Orlando, USA*, 2002.

4. J. Gips, M. Betke, and P. DiMattia. Early experiences using visual tracking for computer access by people with profound physical disabilities. *in Universal Access in HCI: Towards an Information Society for All*, 2001.

5. J. Gips, P. Olivieri, and J.J. Tecce. Direct control of the computer through electrodes placed around the eyes. *Human-Computer Interaction: Applications and Case Studies*, pages 630–635, 1993.

6. R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, UK, 2000.

7. S. Jung and K. Wohn. 3d tracking and motion estimation using hierarchical kalman filter. *IEE Proceedings on Vision, Image and Signal Processing*, 5(144):293–298, 1997.

8. F. Jurie and M. Dhome. A simple and efficient template matching algorithm. *8th International Conference on Computer Vision, Vancouver, Canada*, pages 544–549, 2001.

9. J. Knight, A. Davison, and I. Reid. Towards constant time slam using postponement. *Proceedings of the International Conference on Intelligent Robots and Systems, Maui, USA*, 2001.

10. R. Lockton and A. Fitzgibbon. Real-time gesture recognition using deterministic boosting. *British machine vision conference, Cardiff, UK*, pages 817–826, 2002.

11. G.S. Manku, P. Jain, A. Aggarwal, L. Kumar, and S. Banerjee. Object tracking using affine structure for point correspondences. *IEEE CVPR, San Juan, Puerto Rico*, 1997.

12. F. Quek, T. Mysliwiec, and M. Zhao. Fingermouse: A freehand pointing interface. *Proceedings of the International Workshop on Automatic Face- and Gesture-Recognition, Zurich, Switzerland*, pages 372–377, 1995.

13. L.-Q. Xu, D. Machin, and P. Sheppard. A novel approach to real-time non-intrusive gaze finding. *British machine vision conference, Southampton, UK*, pages 428–437, 1998.