A Handheld Projector Supported by Computer Vision

Kushal, A.; van Baar, J.; Raskar, R.; Beardsley, P.

TR2006-129 January 2006

Abstract

This paper describes the use of computer vision to support the operation of a handheld projector, and describes four applications. Projectors in the past have been used as fixed devices, but the latest generation of 'pocket projectors' is small and portable. We demonstrate the feasibility of using a projector held in the hand, and the types of applications that can be done with a handheld projector. We attach a camera to the projector to support its operation in two ways. Firstly, vision is used to recover the motion of the projector relative to the display surface. A handheld projector with motion recovery allows a range of interesting functionality, and we show how web-browsing can be done with a handheld projector, complete with mouse interaction and text-entry. Secondly, we use the camera to process information about the projection surface - for example we demonstrate an application that allows a user to attach digital information to a physical texture, and later to recover and view the digital data via recognition of the texture.

2006 Asian Conference on Computer Vision (ACCV)

A Handheld Projector supported by Computer Vision

Akash Kushal[1], Jeroen van Baar[2], Ramesh Raskar[2], and Paul Beardsley[2]

[1] Department of Computer Science
University of Illinois, Urbana Champaign, USA
kushal@cs.uiuc.edu
[2] Mitsubishi Electric Research Laboratories, USA
{jeroen,raskar,pab}@merl.com

Abstract. This paper describes the use of computer vision to support the operation of a handheld projector, and describes four applications. Projectors in the past have been used as fixed devices, but the latest generation of 'pocket projectors' is small and portable. We demonstrate the feasibility of using a projector held in the hand, and the types of applications that can be done with a handheld projector.
We attach a camera to the projector to support its operation in two ways. Firstly, vision is used to recover the motion of the projector relative to the display surface. A handheld projector with motion recovery allows a range of interesting functionality, and we show how web-browsing can be done with a handheld projector, complete with mouse interaction and text-entry. Secondly, we use the camera to process information about the projection surface - for example we demonstrate an application that allows a user to attach digital information to a physical texture, and later to recover and view the digital data via recognition of the texture.

1 Introduction

One of the most desirable characteristics for a handheld information device is small size, but the drive to reduce size is in direct conflict with the need for the device's display to be large enough to convey a useful amount of information. An illustration of the problem is that while cellphones are shrinking, they cannot effectively support an application like web-browsing because cellphone screen size is insufficient for web pages that have been created for screens of 15 inches and up. At the same time, projectors are becoming smaller, and several different manufacturers have introduced 'pocket projectors'. Figure 1-left shows a model by Mitsubishi, which weighs 14 oz and fits in the hand. The small size makes it easy to transport, and with a battery life of a few hours, the projector is now becoming a personal portable device just like the laptop.

Now consider a cellphone augmented with a projector. Anticipating that projectors will continue to decrease in size, the device could potentially be very small, but it can create a projection that is similar in size to physical desktop and laptop displays - thus it is possible to have a small device *and* to handle an

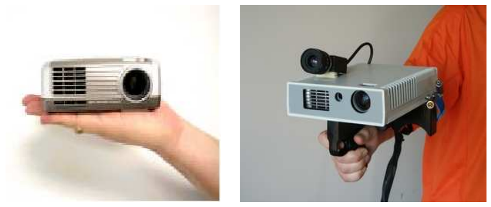

Fig. 1. At left, a commercial 'pocket projector'; at right, our current prototype handheld projector with camera and grip.

information-heavy application like web-browsing. This idea motivates the work here. Of course, a handheld projector must not only display information, but also allow ways to interact (mouse, text-entry). We address both aspects in this paper.

A key technology for a handheld projector is a method for recovering the motion of the projector relative to the display surface. We use computer vision. The primary reason is that vision is versatile, allowing us to develop a range of motion-recovery algorithms for specific applications. In addition, a camera is a cheap component to add to a handheld device. Note that our goal is to create a *self-contained* handheld projector. There are other ways to provide the functionality in this paper if there is fixed infrastructure in the environment (e.g. see [1] for laser pointer interaction in fixed installations). Fixed infrastructure is fine for some applications but our aim is to develop a self-contained device.

Figure 1-right shows the prototype handheld projector. Its components are (a) a Plus V-1080 projector, 1024x768 pixels, 60Hz, (b) a Basler A602F camera, 640x480 pixels, 100Hz, (c) four rigidly attached laser pens, two on either side of the case, (d) a hand-grip on the base with a click button under the index finger for input, (e) umbilical to a computer. The device weighs about 2.5lb so it is heavy for extended use, but it has been suitable for our experiments so far.

Contributions: Previous work on projector-camera interfaces includes [2], and work on steerable projectors includes [3][4]. This paper builds on existing work in [5],[6] and extends it by including (a) handheld projection on display surfaces having an unknown texture, (b) a method for text input and accompanying applications, (c) a 'light stylus' application.

2 Calibration

We do a full euclidean calibration of the projector, camera, and four laser pointers. All references to a 'display surface' below and in the rest of the paper imply a *planar* display surface. The **projector-camera calibration** is as follows

- Camera intrinsics K_c: we take several distinct views of a planar calibration pattern and use a standard calibration method.
- Projector intrinsics K_p: we take several distinct views of a planar calibration pattern while simultaneously using the projector to project a distinct

pattern onto it. The camera simultaneously observes the physical calibration pattern and the projected pattern. Thus we can infer euclidean coordinates for the projected pattern. The problem now reduces to a standard calibration procedure, because we have projector pixel coordinates plus corresponding euclidean world coordinates, for several views of a pattern on a plane. An alternative way to compute projector intrinsics, which avoids the use of a physical calibration pattern, is described in [7].
- Projector-camera extrinsics R, T: we take several distinct views of a blank display surface while using the projector to project a pattern onto it. We collect point correspondences between the camera and projector image planes and compute the fundamental matrix F_{cp} between the camera and projector. We compute the essential matrix $E_{cp} = K_c^t F_{cp} K_p$, and decompose to R, T using a linear computation, followed by a nonlinear computation that minimizes an image plane error [8].

The **laser calibration** is as follows. The laser pens create rays in space. We wish to compute (a) euclidean 3D line equations Q_i of the four laser rays in the projector-camera coordinate frame, (b) for each ray, the projections l_i^c and l_i^p of the ray onto the camera image plane and projector image plane respectively, (c) for each ray, the line homography L_i^{cp} that describes the mapping of points between l_i^c and l_i^p. (The use of this information will be described in subsequent sections). The approach is

- Take an image of a blank display surface while the projector is projecting a pattern, and while the four lasers are projecting four laser spots.
- Detect the pattern, and compute the homography H_{cp} from the camera to the projector image plane.
- Detect the laser spots s_i^c on the camera image plane. For each laser spot, compute $s_i^p = H_{cp} s_i^c$, the inferred position of the laser spot on the projector image plane. Store the correspondence (s_i^c, s_i^p).
- Repeat for three (or more) distinct views.
- Compute the best-fit straight line l_i^c from the stored points s_i^c for the three views. Similarly, compute l_i^p from the stored points s_i^p for the three views. Compute the line homography L_i between l_i^c and l_i^p using the stored correspondences (s_i^c, s_i^p) for the three views.
- Reconstruct the 3D line equations Q_i of each laser ray using the projector-camera calibration and the lines l_i^c and l_i^p.

3 Basic Functions

This section describes the basic functions of the handheld projector - stabilizing a projection so that is fixed on the display surface, our method for doing mouse input with the projector, and use of the lasers to improve the processing.

Stabilizing the Projection: the first goal is to create a stabilized projection (the projection image is fixed on the surface, even when the projector is moving), which is keystone-corrected (the projection image is a rectangle of the desired aspect ratio, even if the projector is skew to the surface). Consider the simplest case. We have four markers $M_1 ... M_4$ on the display surface that form a rectangle with the same aspect ratio as the desired projection image. Call the four vertices of the projection image on the display surface $V_1 ... V_4$. The method to stabilize the projection so that it appears fixed within the marked area is

– Detect $v^c_1 ... v^c_4$, the projections of $V_1 ... V_4$ on the camera image plane. Knowing the corresponding projector pixel coordinates for these vertices, compute H_{cp}, the homography between the camera and projector induced by the display surface.
– Detect $m^c_1 ... m^c_4$, the projections of $M_1 ... M_4$ on the camera image plane.
– Compute $m^p_i = H_{cp} m^c_i$, $i = 1..4$, the inferred projections of $M_1 ... M_4$ on the projector image plane.

Knowing the projection of $M_1 ... M_4$ (the desired projection area) on the projector image plane, it is straightforward to warp the projector image so that it projects to the desired physical position. The whole process is repeated at each new time-step. Figure 2 shows two example time-steps.

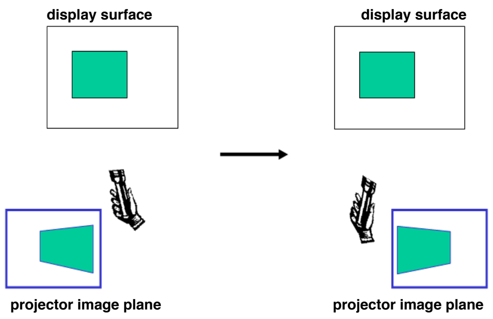

Fig. 2. Stabilization - the projector image plane is continually updated so that the projection on the display surface remains fixed.

Doing Mouse Input with a Projector: we make a single modification to the processing above to do mouse input with the projector - the center pixels of the projector image plane are set at each time-step to show a cursor graphic. The effect of this is that the user sees a stabilized projection on the display surface *plus* a cursor that tracks across the projection in direct correspondence with the projector motion. Once the cursor is at the desired location, items are selected in the usual way by clicking a button on the handheld projector. We can now replicate all the familiar mouse interactions (selecting drop-down menus, clicking buttons, scrolling, dragging) within the projector domain. Figure 3 shows two example time-steps.

We adopt this approach in preference to a touch-pad mouse because (a) a touch-pad would add bulk to the device, (b) it would require two-handed use,

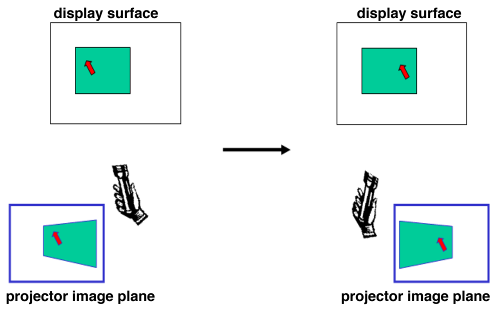

Fig. 3. Mouse interaction - adding the cursor graphic at a fixed position in the center of the projector image plane results in a cursor that tracks across the stabilized projection on the display surface in direct correspondence with the user's pointing motion.

with context switching between the device itself and the display surface, and (c) it is hard to do fine control of a cursor on a large projected area using a small touch-pad.

Using the Lasers to Compute H_{cp}: Part of the processing in the stabilization was to compute $v_1^c ... v_4^c$ with the ultimate goal of computing H_{cp}. But automatically detecting v_i^c (the camera view of the current projection on the display surface) might be unreliable for some projected images. This section describes how we use the (easily detected) laser spots to compute H_{cp}, avoiding the need for v_i^c.

– Detect $x_i^c, i = 1..4$, the projections of the laser spots on the camera image plane. (The projected laser spots x_i^c are constrained to lie on the lines l_i^c that were computed in Section 2, so identifying them is especially easy).
– Compute $x_i^p = L_i x_i^c$, $i = 1..4$, the inferred projections of the laser spots on the projector image plane, where L_i are the line homographies computed in Section 2.
– Compute H_{cp} using the four correspondences (x_i^c, x_i^p).

4 Handling Display Surfaces with Unknown Texture

The previous section described stabilized projection and interaction on a display surface that had known euclidean properties. This section addresses stabilization and interaction on a display surface that has an unknown texture. First assume four distinct points $N_1 ... N_4$ on the display surface. Four points are sufficient to define a projective coordinate frame on the surface. We can define a desired location for the projection in this coordinate frame, and as long as the points are being tracked, we can project to the same fixed position. Furthermore we can readily upgrade to a euclidean coordinate frame (to support keystone-correction) because the handheld projector is calibrated. The approach to initialize the processing is

– Detect $x_i^c, i = 1..4$, the projections of the laser spots on the camera image plane.
– Compute the 3D coordinates of the laser spots, and then compute the 3D coordinates Z for the plane of the display surface.
– Detect n_1^c, n_2^c, the projection on the camera image plane of two arbitrary points N_1, N_2 on the display surface. Backproject n_1^c and intersect with Z to define the origin of the coordinate frame; backproject n_2^c and intersect with Z to define the unit point on the x axis. Hence obtain a euclidean coordinate frame.
– Select the desired location, vertices D_i, for the projection image, in the coordinate frame on the display surface.
– Project D_i to pixel positions d_i^c on the camera image plane.

The approach to propagate the coordinate frame at each time-step, and to display the projection image, is

– Track features $n_i^c, i >= 4$ between the previous frame $j - 1$ and the current frame j, hence obtain feature correspondences between frame 0 and frame j, and compute the homography T_j between the frames 0 and j induced by the display surface.
– Propagate the euclidean coordinate frame to the current camera image by using T_j to transform d_i^c.
– Use H_{cp} to transform d_i^c to the projector image plane. Knowing the coordinates of the desired projection location on the projector image plane, it is straightforward to warp the projector image so that it projects to the desired physical location.

Texture Tracking: Matching between frames employs a global scheme that searches for a consistent transformation over the matched features. The process is initialized with the set of features detected using a Harris corner detector in the base frame (frame 0). For each subsequent frame i we compute the homography T_i between the base frame and the ith frame. To compute T_{i+1} we first transfer all the features in the base frame to frame i using T_i. Then, we search in a small window around each transferred feature for its matching feature in frame $i + 1$. The candidate matches are filtered by an acceptance threshold on the normalized correlation between the matched features in the ith and $i + 1$th frame, and we use RANSAC to identify a set of matches consistent with a homography. The matches are used to compute T_{i+1}. New features that get detected in later frames are transformed from their current frame to base frame and added to the set of base features, to allow the projector to move away from the initial base frame position without losing the tracking.

5 Applications

We propose the following taxonomy of applications for a handheld projector (a) applications that use a display surface where the only texture consists of markers to guide the projection - see Section 5.1, (b) applications that use a display surface with an unknown texture - see Section 5.2 and 5.3, (c) applications that project augmented reality onto a known object - see Section 5.4. There are no quantitative results below. The calibration procedures are a variation on known results for pure camera systems, and there are few observations to be made - the epipolar geometry errors are about 0.5 pixels for the camera-projector and are similar to what one would obtain with two cameras.

5.1 Web-Browsing

Figure 4a shows a projection of a live application - the Google web page. The standard web page is augmented with a 'Text' button at lower-right which the user presses to initiate text-entry. **Figure 4b:** after initiating text-entry, the user holds down the handheld projector's click button and forms a letter 'v'. We use *libstroke* for stroke recognition [9]. **Figure 4c:** after completing the letter, the user releases the click button and is presented with the recognised letter. The same letter is sent to the text-field that is currently active on the web page. After completing text-entry, the user clicks the 'Text' button again to return the mouse to normal cursor mode.

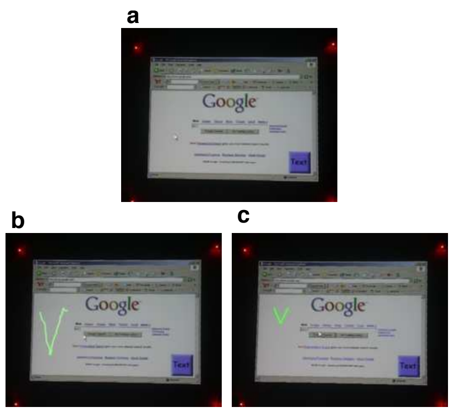

Fig. 4. Web-browsing application including text-entry.

5.2 Light Stylus

Laser pointers are commonly used to indicate a point of interest on a big-screen slide presentation. We extend this functionality to allow a user to create arbitrary doodles on the slides, such as underlines, arrows, or circlings around areas of interest. We call this a 'light stylus'. There are ways to achieve this functionality that make use of fixed infrastructure in the environment, but our approach is a completely self-contained, portable device, making it more flexible in a variety of settings. **Figure 5a** shows an example of a big-screen slide presentation from a fixed projector. **Figure 5b:** the user directs the cursor to the start point of an underline on the text, presses the click button, then directs the cursor to the end point of the underline and double clicks. The underline is subsequently shown at the specified position until the next interaction. **Figure 5c:** the user employs the same interaction with a series of four clicks to draw a box around an object on the slide. **Beware a possible confusion -** the slide presentation is created by a fixed projector, such as one finds in a conference room, *not* by the handheld projector; the handheld projector is used only to create the augmentations in Figures 5b and 5c. This application runs on any texture, not just a slide presentation, and it has also been used to do underlining and circling on posters.

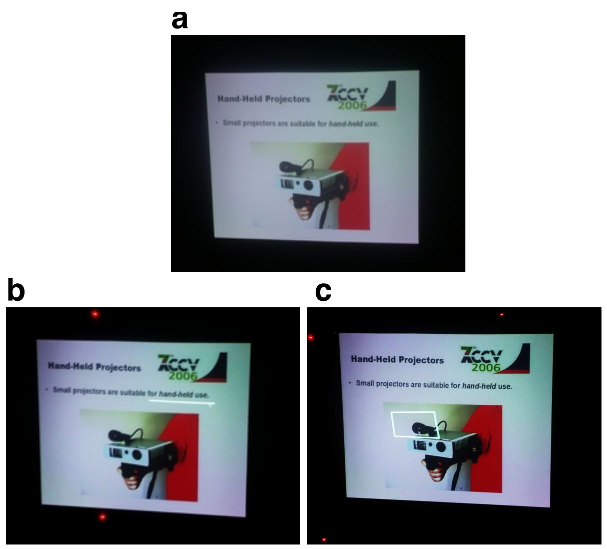

Fig. 5. Light stylus application showing underlining and drawing a box on a slide presentation.

5.3 Electronic Sticky Notes

This application demonstrates how to attach digital information to some physical texture (a CD case), and later retrieve the information automatically the next time that the CD case is seen. **Figure 6a:** the scene consists of some random objects and a CD case. The user directs the cursor to a start point near the CD case, clicks to start a selection, then defines the (projected) blue polygon by clicking at each vertex, and double-clicking at the final vertex. For clarity, we refer to the part of the camera image within the blue polygon as a *texture-key*. The texture-key is stored along with a user-specified text-entry 'Return Date:

25th Jul'. **Figure 6b:** the user directs the handheld projector at a new scene containing the CD case, and requests a retrieve operation. The image is matched against all stored texture-keys. If a match is obtained, the corresponding text-entry is projected next to the recognised object - in this case, the text 'Return Date: 25th Jul' is shown next to the CD case. As an aside, note that projection onto darkish wood is clearly visible. We match the image against stored texture-keys by feature matching, where the features are pixel patches around corners. This is fine for small databases of texture keys and small change in view direction, but of course we would need more sophisticated methods for a truly practical system.

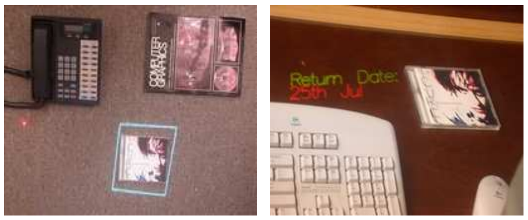

Fig. 6. Electronic sticky note - (a) object selection for attaching a note, (b) retrieval of note.

5.4 Projected Augmented Reality

A key application of the work is projected augmented reality as a way to interface to physical devices that can wirelessly communicate their internal state. The handheld projector retrieves the state, projects it next to the device, allows the user to interact to specify a desired operation, and then transmits the operation back to the device. In this way we can provide complicated control panels for physical devices, without the device needing any sort of physical display or physical input device. See Figure 7 for an example of projected augmented reality.

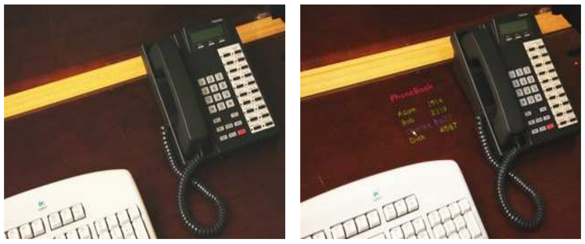

Fig. 7. Projected augmented reality - projecting a phonebook next to a recognised phone.

6 Conclusion

This work is speculative. Based on our experience with a simple game application, which was tried by hundreds of casual users over several days, there is not a problem with its usability. People were able to guide the cursor and to click and drag objects, and while there were comments about the weight of the prototype, many people seemed to feel a real sense of interacting with a projection. But this type of unusual device still raises questions about practicality. Assuming handheld projectors do appear, aren't there easier ways to interact with a projection? It's true that one could attach touchpad or thumbwheels for a familiar type of mouse interaction, but consider how simple and direct the approach in this paper is - one-handed pointing of the projector to guide the mouse. Secondly isn't the current device unwieldy? Our latest generation device has a projector half the weight of the current one, and the trend to more compact projectors is continuing. Isn't the technique wasteful of projector pixels because it uses only part of the projector image plane for the stabilized image? Just as projectors are continuing to become lighter, so the number of pixels continues to increase. And even using a limited part of the projector image plane, we have sufficient pixels for the applications described. In summary, this a workable idea that provides novel functionality for the fast-approaching situation when projectors are incorporated in handheld devices.

References

1. Dan R. Olsen, Jr. and Travis Nielsen, "Laser pointer interaction," in *Proceedings of the SIGCHI conference on Human factors in computing systems.* 2001, pp. 17–22, ACM Press.
2. R. Sukthankar, R.G. Stockton, and M.D. Mullin, "Automatic keystone correction for camera-assisted presentation interfaces," in *Proc. Intl. Conf. Multimedia Interfaces '00*, 2000.
3. Claudio Pinhanez, "Using a steerable projector and a camera to transform surfaces into interactive displays," in *CHI '01: CHI '01 extended abstracts on Human factors in computing systems*, New York, NY, USA, 2001, pp. 369–370, ACM Press.
4. M. Ashdown and Y. Sato, "Steerable projector calibration," in *Proc. IEEE Workshop on Projector-Camera Systems*, 2005.
5. R. Raskar, J. VanBaar, P.A. Beardsley, T. Willwacher, S. Rao, and C. Forlines, "iLamps: Geometrically aware and self-configuring projectors," in *SIGGRAPH 2003 Conference Proceedings*, 2003.
6. Paul A. Beardsley, Jeroen Van Baar, Ramesh Raskar, and Clifton Forlines, "Interaction using a handheld projector," *IEEE Computer Graphics and Applications*, vol. 25, no. 1, pp. 39–43, 2005.
7. T. Okatani and K. Deguchi, "Projector-screen-camera system: Theory and algorithm for screen-to-camera homography estimation," in *Proc. International Conference on Computer Vision*, 2003.
8. R.I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2000.
9. http://www.etla.net/libstroke/, "Libstroke - a stroke translation library," .