PlayAnywhere: A Compact Interactive Tabletop Projection-Vision System

Andrew D. Wilson
Microsoft Research
One Microsoft Way
Redmond, WA
awilson@microsoft.com

ABSTRACT
We introduce PlayAnywhere, a front-projected computer vision-based interactive table system which uses a new commercially available projection technology to obtain a compact, self-contained form factor. PlayAnywhere's configuration addresses installation, calibration, and portability issues that are typical of most vision-based table systems, and thereby is particularly motivated in consumer applications. PlayAnywhere also makes a number of contributions related to image processing techniques for front-projected vision-based table systems, including a shadow-based touch detection algorithm, a fast, simple visual bar code scheme tailored to projection-vision table systems, the ability to continuously track sheets of paper, and an optical flow-based algorithm for the manipulation of onscreen objects that does not rely on fragile tracking algorithms.

Categories and Subject Descriptors: H.5.2 [Information Interfaces and Presentation]: User Interfaces—*Input devices and strategies*

General Terms: Algorithms, Design, Human Factors

INTRODUCTION
The advent of novel sensing and display technology has encouraged the development of a variety of interactive systems which move the input and display capabilities of computing systems on to everyday surfaces such as walls and tables. These efforts are often conducted in the spirit of ubiquitous computing research, where the goal is to make computing resources accessible, seamless, distributed and immediate. The systems pose interesting challenges for interaction design, signal processing and engineering.

Many of these systems are based on the combination of projection for display and computer vision techniques for sensing [4]. In the tradition of direct manipulation and tangible computing, the projection and sensed regions are usually brought into one-to-one correspondence such that the user may interact directly with projected (virtual) objects. The use of computer vision as a sensing technology affords flexibility in sensing a variety of objects placed on the surface, such as hands, game pieces, and so on.

UIST'05, October 23–27, 2005, Seattle, Washington, USA.

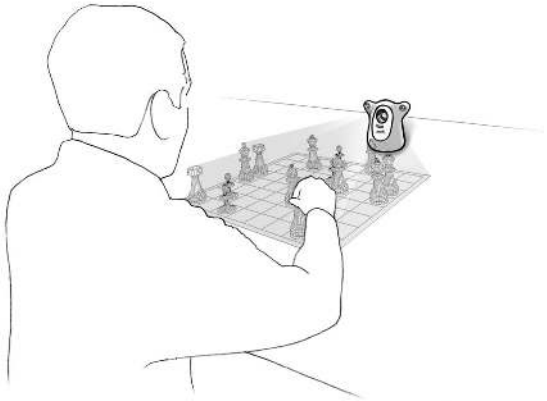

Figure 1: Artist's rendition of a compact tabletop projection and sensing system. Young Kim

Our vision of the future of such devices, illustrated in Figure 1, assumes a continuation of trends in projection and computer vision technology. Here the projector and cameras, as well as computing resources such as CPU and storage, are built into the same compact device. This combined projecting and sensing pod may be quickly placed on any flat surface in the user's environment, and requires no calibration of the projection or sensing system. We believe that portability, ease of installation, and ability to utilize any available surface without calibration are all features required for mainstream consumer acceptance. Imagine a child pulling such a device out of the closet and placing it on a table or the floor of their room to transform the nearby surface into an active play space.

This vision of the future is not so far off. The Canesta projection keyboard [29] and other closely related virtual keyboard devices in many ways resemble our conceptual device. We have in mind however a more general purpose system: one capable of sensing a variety of objects and displaying animated graphics over a large display surface.

In this paper we present the PlayAnywhere prototype, a front-projected computer-vision based interactive table system which uses a new commercially available projection technology to obtain an exceptionally compact, self contained form factor (see Figure 2). It approaches our concept device in that, unlike many other related systems, PlayAnywhere may be quickly set up to operate on any flat

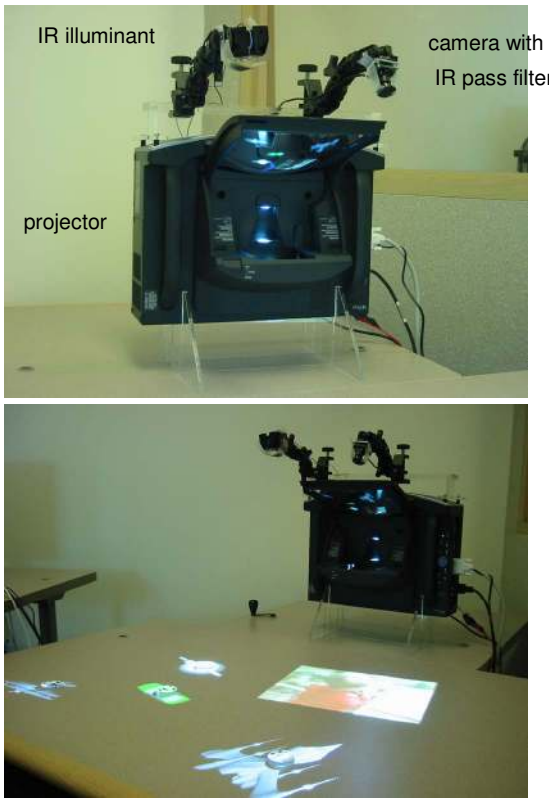

Figure 2: PlayAnywhere prototype (top) consists of WT600 projector on a short pedastal, camera with infrared pass filter and infrared LED illuminant, shown here with heatsink. A circular continuous density filter is applied to the IR illuminant to eliminate hotspots and obtain a more uniform illumination of the surface. PlayAnywhere senses and projects onto a 40" diagonal area (bottom). Here four game pieces and a real piece of paper are detected.

surface, requires no calibration beyond the factory, and is compact while still displaying and sensing over a large surface area. These features make it especially attractive in consumer applications, where distribution, installation, mounting and calibration considerations are paramount. We believe PlayAnywhere to be one of the most practical implementations of the vision-based interactive table idea to date.

PlayAnywhere demonstrates a number of important sensing capabilities that exploit the flexibility of computer vision techniques. We introduce a touch detection algorithm based on the observation of shadows, a fast, simple visual code format and detection algorithm, the ability to continuously track sheets of paper, and finally, an optical flow-based algorithm for the manipulation of onscreen objects that does not rely on fragile tracking algorithms.

RELATED WORK

There has been a great variety of interactive table and wall research prototype systems. Here we limit discussion to *imaging touch screens*, those that utilize an image or image-like representation which indicates the presence and possibly the appearance of multiple objects placed on a surface.

One popular approach is to mount a camera and projector high on a shelf or on the ceiling [3, 11, 15, 25, 32, 33]. Such mounting configurations are typically necessary because of the throw requirements of projectors and the typical focal length of video cameras. Such a configuration has the following drawbacks:

- Ceiling installation of a heavy projector is difficult, dangerous, requires special mounting hardware. and is best left to professionals.
- Once the installation is complete, the system and the projection surface cannot be moved easily.
- Often minor vibrations present in buildings can create problems during operation and make it difficult to maintain calibration [33].
- The user's own head and hands can occlude the projected image as they interact with the system. To our knowledge, however, there has been no systematic analysis of the true impact of such occlusions.

A second approach is to place the projector and camera behind a diffuse projection screen [14, 16, 20, 31]. While this enables the construction of a self-contained device, allows the placement of codes on the bottom of the objects, and eliminates occlusion problems, this approach also has drawbacks:

- It is difficult to construct such a table system with large display area which also allows users room enough to put their legs under the table surface.
- Because the camera is looking through a diffuse surface, the imaging resolution is limited (though see [35] for one way to address this problem). High resolution capture of documents, for example, is impossible.
- A dedicated surface is required, and the resulting housing for the projector and camera can be quite large.

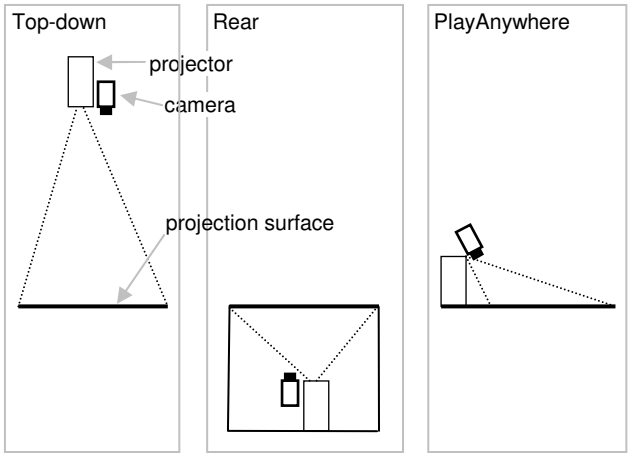

Figure 3: Most projection-vision systems either employ front projection with projector and camera mounted above (left), or rear projection with projector and camera in a cabinet (middle). PlayAnywhere employs a camera and projector sitting off to the side of the active surface (right).

This presents manufacturing and distribution problems for a real product.

Front and rear projection-vision system configurations are illustrated diagrammatically in Figure 3.

Finally, there are a number of systems which embed sensing electronics into the surface itself [6, 24]. These systems typically result in very fast and precise detection of touch compared to vision based approaches, but lack much of the flexibility in terms of other objects to be sensed. Others support only objects with special embedded hardware devices and do not detect touch [23]. These systems usually rely on overhead projection.

Computer vision-based tables are capable of interesting sensing capabilities, including detection and recognition of objects place on the surface. In this paper we present novel techniques to enable a variety of sensing capabilities and interactions, many of these capabilities have been studied in previous work.

2D visual codes are often used as tags to identify physical objects in augmented reality scenarios [7, 11, 22, 26]. For example, a printed page can be augmented with a visual code which enables the system to call up the corresponding electronic form.

Robust finger tracking has been studied in the context of table systems [13, 15, 17, 38], but generally 'clicking' or 'pen down' is implemented by dwelling or other gesture recognition. True detection of touch can be detected roughly with two cameras [5, 19, 35, 36]. In the present work, we explore the analysis of shadows to detect touch and infer hover height. A related formulation uses shadows to infer the height of objects above a surface but is unsuited to the case where the object is touching the surface and so occludes its own shadow [28], while another approach using observing shadows using an illuminant coaxial with the camera is unable to infer precise depth or hover information [30].

Finally, there is interest in detecting real printed paper pages, and how interactive systems may be integrated with the world of real paper documents [12, 14, 15, 21, 27, 33]. In the present work we will consider the precise real time continuous tracking of printed pages placed on the surface.

PLAYANYWHERE CONFIGURATION

Figure 2 shows the PlayAnywhere prototype, which includes a projector, camera and infrared illuminant assembled as a single piece designed to sit on a flat surface such as a table, desk, or floor. In the following, we detail each of these components.

Projector

PlayAnywhere uses a NEC WT600 DLP projector to project a 40" diagonal image onto an ordinary table surface. The NEC WT600 is an unusual projector in that it uses four aspheric mirrors (no lenses) to project a normal 1024x768 rectangular image from a very oblique angle, and at extremely short distance. For a 40" diagonal image, the WT600 requires 2.5" between its leading face and the projection surface, while a 100" diagonal image is obtained at a distance of 26". These characteristics make it very well suited for PlayAnywhere, in that it allows for the projector to sit directly on the projection surface (on a short pedestal), rather than hang suspended over the surface.

The application of this projector has a number of advantages:

- Difficult and dangerous overhead installation of the projector is avoided.
- It is reasonable to assume that the plane of the surface holding the projector is the projection plane. If the camera and illuminant is rigidly mounted to the projector, there is no need to re-calibrate the camera and projection to the surface when the unit is moved. Similarly, since the height of the camera and projector above the surface is constant, there are no problems related to adjusting focal length of either the camera or projector when the unit is moved.
- With the oblique projection, occlusion problems typical of front-projected systems are minimized. For example, it is possible for the user to stand over PlayAnywhere without their head occluding the projected image.
- A 40" diagonal projection surface is adequate for many advanced interactive table applications, including complex gaming scenarios that go beyond simple board games, and manipulation of multiple photos, printed pages, etc.

Disadvantages of this projector include:

- By not controlling the projection surface, the image projection quality cannot be guaranteed.
- While the front projection arrangement allows users to sit comfortably with their legs under the table, one side of the table is effectively blocked by the projector.

Camera and Illuminant

As in other projection-vision systems, we illuminate the scene with an infrared source and block all but infrared light to the camera with an infrared-pass filter. This effectively removes the projected image from the scene.

The PlayAnywhere projector provides a natural place to mount one or more cameras and an infrared illuminant. By rigidly mounting the cameras and illuminant to the projector, the calibration of the vision system to the display is the same regardless of where PlayAnywhere is situated, and may be determined at the factory.

One method to perform sensing and detection of objects on the surface is to use two cameras and simple stereo calculations (as in [5, 19, 35, 36]), but with PlayAnywhere we chose to instead use one camera and explore simple image techniques that allow touch detection by examining the shadows of objects, detailed later. We place the IR illuminant off axis from the single camera so that objects above the surface generate controlled shadows indicating height.

PlayAnywhere uses an Opto Technology OTLH-0070-IR high power LED package and a Sony ExView analog gray-

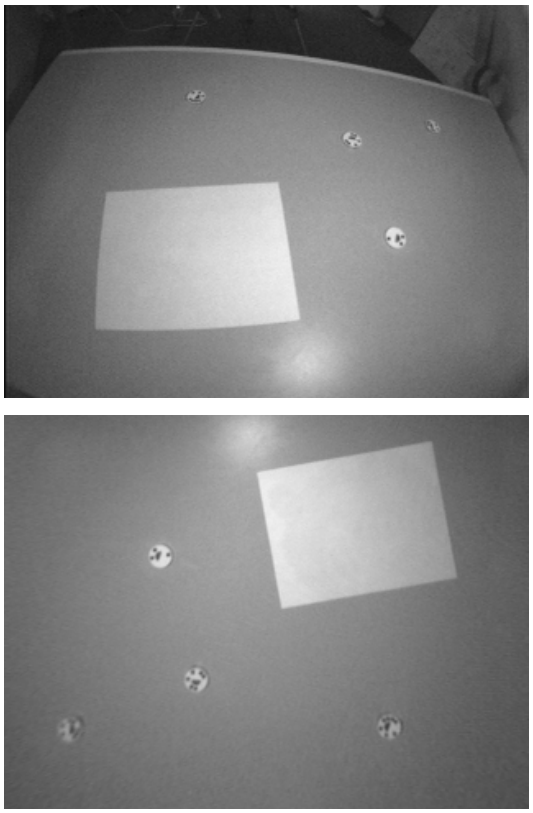

Figure 5: Initial image processing removes lens distortion effects from input image (top) and matches the image to the display. The rectified image (bottom) is registered with the displayed image.

scale CCD NTSC camera. The ExView was chosen for its high sensitivity in the near infrared domain. To minimize the size of the overall package, the camera is mounted near the top of the projector, giving an oblique view of the surface. A very wide angle micro lens (2.9mm focal length) is suitable to capture the entire projected surface.

In future prototypes it may be possible to avoid such an oblique camera view by using a shift lens configuration (as employed by most conventional projectors) or by embedding the camera with the projector in such a way that they share the same optical path.

IMAGE PROCESSING

In this section we describe the image processing and computer vision capabilities of PlayAnywhere, including basic image processing techniques that are common to projection vision systems, a novel method of detecting touch, a simple visual codes format and a demonstration of continuous tracking of paper pages. We finally introduce a novel method of manipulating onscreen objects using optical flow techniques.

Image Rectification

The wide angle lens imparts significant barrel distortion on the input image, while the oblique position of the camera imparts a projective distortion or foreshortening. In the initial image processing step of PlayAnywhere, an image rectification process removes both distortions via standard bilinear interpolation techniques. Parameters necessary to correct for lens distortion are recovered by an off-line procedure [10]. The projective transform is determined by finding each of the four corners of the projected display by placing infrared reflective markers (paper) on the surface at known locations indicated by the projected image. Image rectification is illustrated in Figure 5. Note that due to the configuration of PlayAnywhere and the assumption that the unit sits on the projection plane, this calibration step need not be performed again when the unit is moved.

With image rectification, the input image and projected image are brought into one to one correspondence; i.e. a rectangular object on the surface appears as a rectangular object in the image at the same (scaled) coordinates. One limitation of this process is that, due to the oblique view of the camera, objects further away from the unit will appear at a lower resolution. Consequently, the minimum effective resolution on the surface is less than that of the acquired image (640x480 pixels).

Touch and Hover

For PlayAnywhere we are interested in methods to detect touching the surface without relying on special instrumentation of the surface, so that the device may operate on any available flat surface. One approach is to project a sheet of infrared light just above the surface and watch for fingers intercepting the light from just above, much as with the Canesta device [29]. Here we present a technique which exploits the change in appearance of shadows as an object approaches the surface.

Figure 4 shows the (rectified) input image with two hands

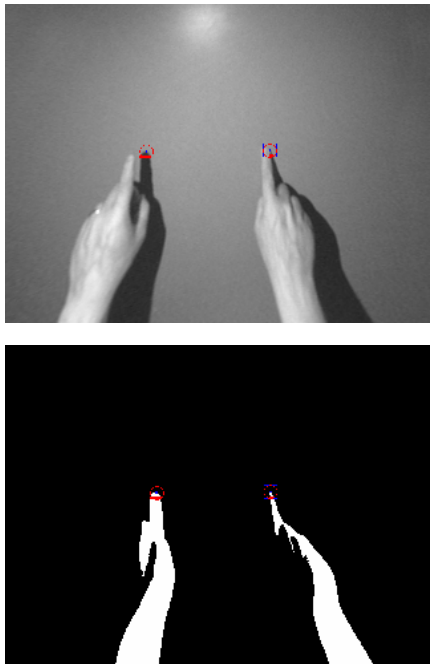

Figure 4: Finger tracking and touch detection is based on shadow shape analysis. The input image (top) shows the left hand above the surface and the right hand (index finger) touching the surface. The image is first binarized to recover an image which contains only shadows (bottom).

in the scene. The hand on the left is a few inches above the surface, while the index finger of the hand on the right is touching the table surface. Note that as the index finger approaches the surface, the image of the finger and its shadow come together, with the finger ultimately obscuring the shadow entirely where it is on the surface. Because the illuminant is fixed with respect to the camera and surface, it should be possible to calculate the exact height of the finger over the surface if the finger and its shadow are matched to each other and tracked. This height could be used as a hover signal for cursor control, or 3D cursor control.

In our current approach, we avoid tracking the finger because to do so would require making some assumptions about the appearance of the surface and fingers (for example, that fingers and the surface have significantly different brightness), and instead focus on analysis of the shadow. Recovering the shadow reliably requires only that the surface reflect infrared and that the device's infrared illuminant is significantly brighter than stray infrared in the environment. Both assumptions are reasonable given that the user is likely to place the device on a surface where the projection has good contrast and brightness (i.e., not on a black surface, or in a very bright room).

A shadow image can be computed from the rectified input by a simple thresholding operation (see Figure 4). Candidate finger positions are generated by finding the highest (closest to the device) point on each of the distinct shadows in the image which enter the scene from the bottom of the image (away from the device). These conditions typically yield a candidate for the most forward finger of each hand on the surface, if the user is reaching in from the front, and rejects other objects on the surface that may generate their own shadows. Such finger candidates may be found quickly by computing the connected components of the smoothed shadow image.

Whether the finger is touching the table may be determined by simple analysis of the shape of the shadow. Figure 7 shows the shadow at a finger tip for a finger on and off the surface. In the current implementation, we simply threshold the width of the finger shadow computed at a location slightly below the topmost point. In the future, this detection algorithm should be augmented by a verification algorithm (e.g., [17]), but in our experience, the provision that the candidate finger must lie on a shadow that extends to the bottom of the image tends to limit false positives if there are few other physical objects on the table surface. Objects that are on the table can be considered part of the shadow if they are particularly dark, and can corrupt touch detection if they are nearby. Pointy dark objects are likely to generate false positives only if they extend to the bottom of the image and thus mimic arm shadows.

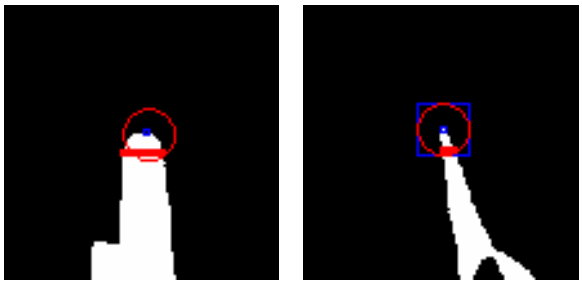

Figure 7: Touch is determined by simple shape heuristics. A finger on the surface occludes almost all of its own shadow (right), while a finger above the surface does not (left).

Presently, this approach recovers only one finger per hand. More sophisticated finger shape analysis can be used to recover multiple fingers per hand perhaps at some cost in robustness. Because very few assumptions about the shape of the hand are made, the pose of hand is not critical, and so the hand can be relaxed.

The precision of touch location is limited by the resolution of the imaged surface, which has been subjectively estimated with grating charts to be about 3-4mm (approximately 4.5 image pixels). Simple trigonometry can be used to show that this spatial resolution implies a roughly equal resolution in the determination of height and therefore touch accuracy by the method described above. This agrees with the subjective experience of using the system.

While the touch location precision is not quite enough to support traditional GUI mouse-based interaction, we have implemented buttons using this finger detection scheme and a simple drawing application, illustrated in Figure 6. We have also begun experimenting with TabletPC integration to incorporate its various text entry methods, tap-and-hold for right-click model, and use of hover.

PlayAnywhere Visual Code

Visual codes have been applied in various augmented reality and table scenarios, where they can be used to identify potentially any object large enough to bear the code without recourse to complex generalized object recognition techniques. In tabletop scenarios, such visual codes are especially useful to locate and identify game pieces, printed pages, media containers, knobs and other objects that are generic in appearance but vary in application semantics. As a knob, for example, an identified piece could adjust the color and contrast of a digital photo.

A number visual code schemes are used in augmented real-

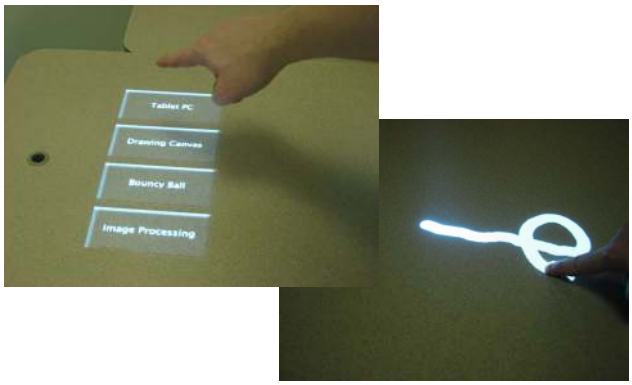

Figure 6: PlayAnywhere can detect hover and touch. Left: Buttons appear when a finger hovers over the upper left corner of this application. Touch presses the button. Right: A simple touch-based drawing application.

ity research (e.g., [7]). He we outline a code format and algorithm designed for PlayAnywhere that is particularly fast and simple to implement (in fact it is intended to be implemented on today's GPU hardware), and requires no search to determine code orientation.

Generally the problem of designing a code format is one of balancing the opposing goals of obtaining a simple detection algorithm that works with the various transformations observed (e.g., translation, rotation) while supporting a useful number of code bits (see [22] for an interesting discussion). In the case of calibrated tabletop vision systems such as PlayAnywhere, where we may be interested in only game pieces on the surface, for example, we can assume that each instance of the code appears in the image with known, fixed dimensions, thus simplifying the recognition and decoding process.

The design of the PlayAnywhere code, illustrated in Figure 8, was driven from two observations. First, the presence and orientation of strong edges in the image may be computed using simple, fast image processing techniques such as the Sobel filter [8]. Thus, if the code has a distinct edge as part of the design, the orientation of that edge can determine the orientation of the instance of the code. Secondly, if the code design supports significantly more bits than is needed for the application (e.g., an application may require only 12 unique code values, one for each of the game piece types in chess, but the 12 bit code supports 4,096 unique codes values), then the code values may be chosen such that if one is found through any process, we are willing to take it as an indication of a valid instance of the code. These two observations used together make for a very simple detection algorithm, as follows.

1. Compute the edge intensity and orientation everywhere in the image using the Sobel filter.
2. For each pixel with sufficiently high edge intensity, use the edge orientation to establish a rotated local coordinate system.
    a. In the rotated coordinate system, read each pixel value in the rectified image corresponding to each bit in the code according to the code layout. Threshold each based on the minimum and maximum value read, to arrive at a code value.
    b. Check the code value against a table of codes used in the current application. We have a candidate instance if there is a match.
3. Rank each candidate according to some criteria (e.g., difference between maximum and minimum pixel values read). Iterate until no more candidates: Take top ranked candidate as valid code instance, and eliminate remaining candidates that overlap.

In practice, depending on the code bit depth, the number of application code values required and the nature of potential distracters in the image, it may be necessary to add a further step that verifies the instance. For example, in the present implementation we limit consideration to image locations that appear to be the center of circular contours of the

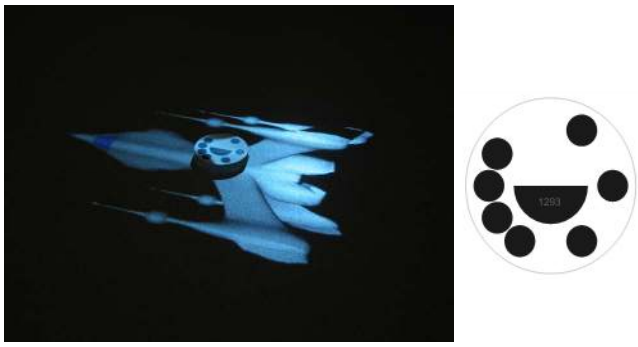

Figure 8: 3D graphics model is projected onto an identified game piece (left) with orientation determined by strong edge in the center of the pattern, and 12 bit code given by pattern around the edge (right). Each game piece is 1.4" in diameter, printed on a laser printer and glued to an acrylic plastic disc of the same diameter.

game piece diameter. Such contours can be found quickly using the Hough transform [1] applied to circles, reusing the edge orientation information computed above: a 2D histogram (image) representing the presence of circles centered at point is created by, for each pixel in the input image, calculating the center of the circle of a given radius and edge orientation found at the input coordinates, and incrementing the histogram at the calculated center. Each point in the resulting histogram indicates the likelihood of a circle of the given radius centered there.

We've found the resulting implementation to be a good balance between simplicity of design and performance, but have not rigorously evaluated the diagnostic power and robustness of the algorithm or explored optimizations on the basic design. Figure 8 illustrates the PlayAnywhere visual code. One limitation of this scheme is that the user's hand can occlude a visual code. Without hysteresis or integration with the shadow-based touch algorithm, the system will conclude that the piece has disappeared.

Page Tracking

Systems such as PlayAnywhere are natural platforms for exploring ways to blur the boundary between the virtual, electronic office document and the real thing, as well as scenarios that exploit the natural and familiar feel of manipulating and drawing on paper.

PlayAnywhere's real time page detection and tracking permits the user to move and rotate virtual objects with the same ease as manipulating a printed page on a desk. Ultimately, this capability can support more complex scenarios. For example, consider making a real charcoal drawing on paper. This drawing could then be captured to an image precisely using the page tracking information, and then later projected back onto the surface as a virtual object, or even onto a blank piece of paper, or another work in progress.

PlayAnywhere's page tracking algorithm is based on a Hough transform with the Sobel edge and orientation information as input. This gives a histogram over orientation and perpendicular distance to the origin which indicates the presence of strong lines in the image. Given the dimensions

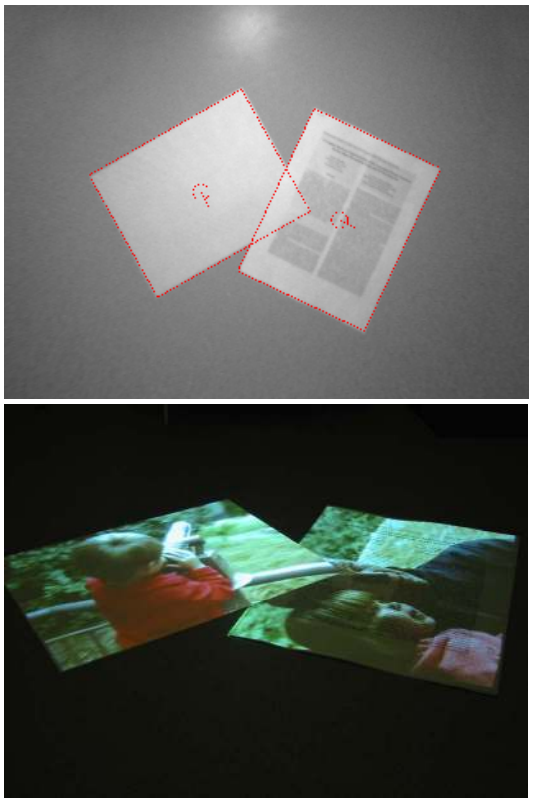

Figure 9: PlayAnywhere's page tracking capability detects two overlapping pages placed on the surface (top). On the table, videos are projected precisely on the printed pages (bottom).

of a page size to detect, it is straightforward to find appropriate pairs of parallel lines a set distance apart. Two pair of parallel lines perpendicular to each other is verified as a page by ensuring that there are strong edges along a significant fraction of the lines in the original Sobel image. This proportion can be tuned to allow for pages to overlap.

With this algorithm, multiple paper pages of known dimensions may be continuously tracked by PlayAnywhere with enough precision to project a virtual image on the page as it is moved around the surface. Presently multiple pages are tracked and disambiguated purely by assuming small frame to frame movement not page appearance. This tracking process also allows for pages to be turned 180 degrees recognizably. Multiple (known) page sizes may also be simultaneously detected with minimal additional computation. Figure 9 shows page detection results and its application to the projection of video onto physical pages.

Flow Move

Multiple touch interfaces such as PlayAnywhere naturally support a direct manipulation style of interacting with virtual objects where the user can initiate interacting with the object by touch and natural gesture. For example, in a photo browsing and sorting application, it is natural and convenient to move and rotate virtual photos as if they were real photos lying on the surface, and to support other operations that may be physically impossible but desirable and plausible anyway, such as resizing.

One approach to implementing an interaction that allows translation, rotation and scaling of an onscreen object is to track one or more parts of one or more hands placed on the virtual object, and then continuously calculate the change in position, orientation and scale based on the relative motion of those tracked points. In the case of only one tracked point, only translation is supported. Such an algorithm presumes that points touching the surface are tracked reliably over time. In the case of imaging based touch systems, this requirement can raise thorny ontological questions that simple pattern recognition techniques are ill-equipped to handle: are those two blobs distinct fingers moving away from each other, or two parts of the same finger seen in the last frame? In practice, this approach requires that the user handle the system in a way that the input may be interpreted unambiguously. For example, the user may find that such a system works better if they user only the tips of two fingers.

A second approach is to augment the presentation with onscreen widgets and modal interactions that channel the user into providing unambiguous input [37]. This has the advantage in that, as with modern GUIs, the range of possible interactions is limited only by the designer's imagination. With widgets and modal interactions, it is also possible to manipulate the object very precisely. The drawback is that the user is required to discover and learn the operation of these devices, which may not behave analogously to the

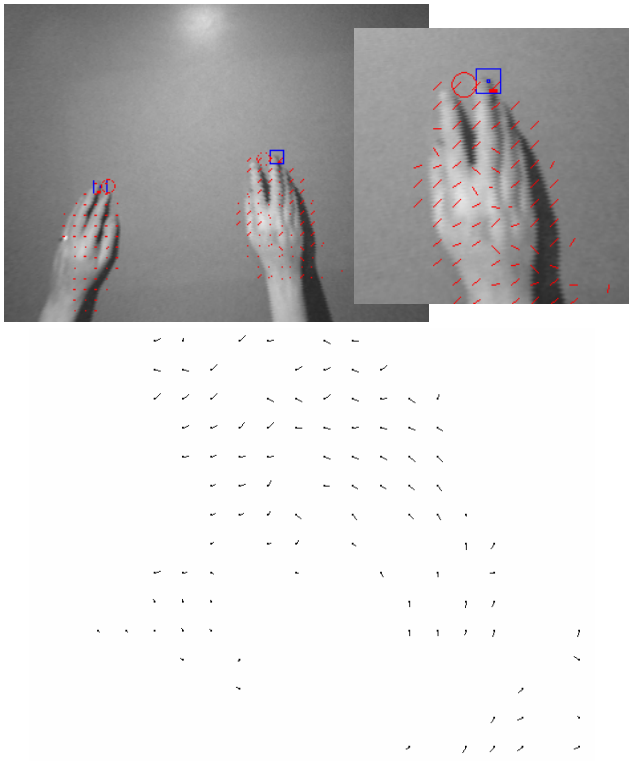

Figure 10: Optical flow field is illustrated for two hands moving apart to scale an image larger (top). Circles and squares indicate points of touch. While each motion estimate may be noisy, taken together (over both hands), the overall motion estimate is less noisy. A single hand can effect rotation (bottom).

interaction they are meant to replace (e.g., menus). This approach can be disappointing from a more philosophical perspective, since to some extent it compromises the natural, immediate and modeless esthetic that is the main strength of interactive table systems in the first place.

With PlayAnywhere we have explored an alternative approach to the simultaneous translation, scaling and rotation of virtual objects based on optical flow techniques that avoids tracking issues and does not rely on onscreen widgets. This is alluded to in our previous work [34].

Optical flow computations make very few assumptions about the nature of the input images and typically only require that there be sufficient local texture on the moving object [2]. We avoid the difficulties of developing a reliable absolute position-based tracker by computing simple statistics that summarize the flow field and restrict ourselves to computing relative motion information.

Optical flow is computed from the most recently acquired image I_t and the previous image I_{t-1}. Our prototype uses a simple block matching technique in which, for each point (x, y) on a regular grid in the image, the integer vector quantity (dx, dy) is determined such that the image patch centered on (x, y) at time $t-1$ most closely matches the image patch centered on $(x+dx, y+dy)$ at time t. In this calculation, image patches are compared by computing the sum of pixelwise absolute differences (low values indicate close match). For a given patch in the image, we select (dx, dy) that minimizes

$$\sum_{x,y \in \text{patch}} \left| I_{t-1}(x, y) - I_t(x+dx, y+dy) \right|$$

An onscreen object may be simultaneously rotated, scaled and translated in 2D by constructing the transform matrix for a graphics object as the composition of rotation, scaling and translation parameters. In the appendix we present a mathematical framework for computing the simultaneous rotation, scaling and translation of the hand by computations on the flow field. If an application requires it, one or more of the three transforms may be ignored. For example, general object manipulation in an image manipulation program may make use of all transforms simultaneously (see Figure 11), while rotating an onscreen knob may require only rotation information.

Figure 11: Flow field-based manipulation of objects applied to panning, rotating, and zooming a high resolution map.

It is probably desirable to manipulate the object only when the user touches the object. Flow-based movement can be enabled for a given object at a high level if any touch is detected by the shadow-based process described above, or alternatively, if there is any motion observed on a patch of the image that lies on the virtual object that is not a shadow, and which has no corresponding shadow.

Because the flow-based technique does not rely on fragile, narrow models of detection and tracking, it supports a variety of usage styles. For example, it is possible to rotate an object by placing the hand on the object and rotating the whole hand. The same rotation can be made by putting two hands on either side of the object and moving them about a center of rotation. Scaling can similarly be accomplished by spreading or contracting a single hand, or by moving two hands closer or further apart. Subjectively, the interaction is fluid and reminiscent of a simulation of the physics of moving a stretchy piece of paper. Figure 10 shows input images and flow fields for two hands moving apart, and the flow field for a single rotating hand.

One of the limitations of the flow-based movement technique is that because each flow field encodes only relative motion, errors in the flow field computation can accumulate over many frames. Ultimately, motion may not appear to integrate correctly: i.e., if the hand is moved from one location to another and back to the original location, the virtual object may not be placed its original location exactly. This is a common problem in flow-based vision techniques and can be solved in a number of ways, but informal observation suggests that users are fairly adept at exploiting the continuous visual feedback to get the desired motion and will tolerate small amounts of drift.

It may be advantageous in some situations to perform motions that are not mapped exactly from the input. For example, the absolute amount of texture (number of flow vectors) may be used to control an acceleration profile that enables precise rotation when only one hand is used, while large, fast rotations may be made with two hands.

COMPUTER VISION PROS AND CONS

One of the primary advantages of using a projection and vision system for interactive tables is the extreme flexibility afforded by computer vision processes. For example, with the page tracking algorithm described above and generic object recognition algorithm such as SIFT [18] it is a straightforward task to implement a system which recognizes which of several known audio CDs is placed on the table surface. Visually coded knob pieces placed on the surface can control volume or some other aspect of the CD's playback, while a left or right gesture with the hand can cause the player to go to the next or previous track.

This flexibility comes at some cost:

- Computer vision techniques have a high computational cost. Many image processing techniques can exploit GPU hardware with some effort, but for higher frame rates, computational cost will be a concern for some time (the present system achieves 30Hz frame rate on commodity hardware with all features enabled, no GPU use but high CPU consumption).
- Most commodity camera systems acquire images at only 30Hz, which is not fast enough to support certain kinds of high fidelity input. For example, the TabletPC stylus gives reports at 133Hz. More recent CMOS cameras support frame rates in (inverse) proportion to the size of region of interest, so there is some potential for increasing the frame rate for some applications. Furthermore, the typical resolutions of today's video cameras make them unsuitable for fine work.
- Often a sensing system tailored to a particular task will give better performance than a general purpose vision system. For example, the touch algorithm described in the present work will sometimes report touch when the finger is some millimeters above the surface. Touch sensing alone is probably better implemented with something like the Canesta device, or in the case that the surface may be instrumented, a capacitance-based system such as SmartSkin or DiamondTouch.

Ultimately, the vision-based approach makes sense only if the flexibility available in this approach is used. This in turn depends on the various applications we dream up—if all we intend is to emulate the single cursor in Windows on a table, the computer vision approach is probably unnecessarily complex and inferior in many respects. However, as demonstrated by PlayAnywhere and other systems, computer vision-based systems can support many scenarios that go beyond the usual GUI, including the merging of paper and virtual documents, gaming, collaborative and other multi-touch applications, image-based artistic applications, and a variety of scenarios related to ubiquitous computing and tangible interfaces.

CONCLUSION

We have introduced PlayAnywhere, an interactive projection vision system with a unique form factor, and a number of sensing capabilities that demonstrate the flexibility of computer vision-based tables. PlayAnywhere suggests a a form factor that is in many ways more attractive than either rear-projection systems or front projected systems to date.

REFERENCES

1. Ballard, D., and C. Brown *Computer Vision*. Prentice-Hall, 1982.
2. Barron, J., D. Fleet, S. Beauchemin, and T. Burkitt, Performance of Optical Flow Techniques. in *Computer Vision and Pattern Recognition*, (1992), 236-242.
3. Berard, F., The Magic Table: Computer Vision Based Augmentation of a Whiteboard for Creative Meetings. in *IEEE International Conference in Computer Vision, Workshop on Projector-Camera Systems (PROCAMS'03)*, (2003).
4. Buxton, W. Projection-Vision Systems: Towards a Human-Centric Taxonomy *Unpublished Manuscript*, Toronto: Buxton Design, 2005.
5. Corso, J., D. Burschka and G. Hager, The 4D Touchpad: Unencumbered HCI with VICs. in *Proceedings of the CVPR-HCI Workshop*, (2003).
6. Dietz, P.H., and D. L. Leigh, DiamondTouch: A Multi-User Touch Technology. in *ACM Symposium on User Interface Software and Technology*, (2001), 219-226.
7. Fiala, M. ARTag Revision 1, A Fiducial Marker System Using Digital Techniques *NRC Technical Report (NRC 47419)*, National Research Council of Canada, 2004.
8. Gonzalez, R., and R. Woods *Digital Image Processing: Second Edition*. Prentice-Hall, 2002.
9. Horn, B.K.P. Closed Form Solution of Absolute Orientation Using Unit Quaternions. *Journal of the Optical Society*, *4* (4). 629-642.
10. Kang, S.B. Radial Distortion Snakes. *IEICE Transactions on Information and Systems*, *E84-D* (12). 1603-1611.
11. Kato, H., M. Billinghurst, I. Poupyrev, K. Imamoto, K. Tachibana, Virtual Object Manipulation on a Table-Top AR Environment. in *Proceedings of ISAR 2000*, (2000).
12. Kim, J., S. Seitz, and M. Agrawala, Video-based Document Tracking: Unifying Your Physical and Electronic Desktops. in *Proceedings of ACM Symposium on User Interface Software and Technology*, (2004), 99-107.
13. Kjeldsen, R., C. Pinhanez, G. Pingali, and J. Hartman, Interacting with Steerable Projected Displays. in *International Conference on Automatic Face and Gesture Recognition*, (2002).
14. Klemmer, S.R., M. W. Newman, R. Farrell, M. Bilezikjian, J. A. Landay, The Designer's Output: A Tangible Interface for Collaborative Web Site Design. in *ACM Symposium on User Interface Software and Technology*, (2001), 1-10.
15. Koike, H., and Y. Sato, Y. Kobayashi Integrating Paper and Digital Information on EnhancedDesk: a Method for Realtime Finger Tracking on an Augmented Desk System. *ACM Transactions on Computer-Human Interaction (TOCHI)*, *8* (4). 307-322.
16. Leibe, B., T. Starner, W. Ribarsky, Z. Wartell, D. Krum, J. Weeks, B. Singletary, and L. Hodges. Toward Spontaneous Interaction with the Perceptive Workbench. *IEEE Computer Graphics and Applications*, *20* (6). 54-65.
17. Letessier, J., and F. Berard, Visual Tracking of Bare Fingers for Interactive Surfaces. in *ACM Symposium on User Interface Software and Technology*, (2004).
18. Lowe, D. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, *60* (2). 91-110.
19. Malik, S., and J. Laszlo, Visual Touchpad: A Two-Handed Gestural Input Device. in *Proceedings of the International Conference on Multimodal Interfaces*, (2004), 289-296.

20. Matsushita, N., J. Rekimoto, HoloWall: Designing a Finger, Hand, Body and Object Sensitive Wall. in *ACM Symposium on User Interface Software and Technology (UIST)*, (1997).
21. Newman, W., and P. Wellner, A Desk Supporting Computer-based Interaction with Paper Documents. in *CHI '92*, (1992).
22. Owen, C., F. Xiao, and P. Middlin, What is the Best Fiducial? in *Augmented Reality Toolkit, the First IEEE International Workshop*, (2002).
23. Patten, J., H. Ishii, J. Hines, G. Pangaro, Sensetable: A Wireless Object Tracking Platform for Tangible User Interfaces. in *Proceedings of CHI 2001*, (2001), 253-260.
24. Rekimoto, J., SmartSkin: An Infrastructure for Freehand Manipulation on Interactive Surfaces. in *Proc. CHI 2002*, (2002), 113-120.
25. Rekimoto, J., and M. Saitoh, Augmented Surfaces: A Spatially Continuous Workspace for Hybrid Computing Environments. in *Proceedings of CHI 99*, (1999), 378-385.
26. Rekimoto, J., and Y. Ayatsuka, CyberCode: Designing Augmented Reality Environments with Visual Tags. in *Designing Augmented Reality Environments (DARE 2000)*, (2000).
27. Robinson, J., and C. Robertson, The LivePaper System: Augmenting Paper on and Enhanced Tabletop. in *Computers and Graphics*, (2001), 731-743.
28. Segen, J., and S. Kumar, Shadow Gestures: 3D Hand Pose Estimation Using a Single Camera. in *Proceedings of Computer Vision and Pattern Recognition*, (1999), 479-485.
29. Tomasi, C., A. Rafii, and I. Torunoglu Full-size Projection Keyboard for Handheld Devices. *Communications of the ACM*, *46* (7). 70-75.
30. Ukita, N., and M. Kidode, Wearable Virtual Tablet: Fingertip Drawing on a Portable Plane-object Using An Active-Infrared Camera. in *Proceedings of Intelligent User Interfaces*, (2004), 169-176.
31. Ullmer, B., H. Ishii, The metaDESK: Models and Prototypes for Tangible User Interfaces. in *ACM Symposium on User Interface Software and Technology*, (1997), 223-232.
32. Underkoffler, J., and H. Ishii, Illuminating Light: An Optical Design Tool with a Luminous-Tangible Interface. in *Proceedings of the ACM CHI 98 Human Factors in Computing Systems Conference*, (1998), 542-549.
33. Wellner, P. Interacting with Paper on the DigitalDesk. *Communications of the ACM*, *36* (7). 86-97.
34. Wilson, A., FlowMouse: A Computer Vision-Based Pointing and Gesture Input Device. in *Interact '05 (to appear)*, (2005).
35. Wilson, A., TouchLight: An Imaging Touch Screen and Display for Gesture-Based Interaction. in *International Conference on Multimodal Interfaces*, (2004).
36. Wren, C.R., Y. A. Ivanov, Volumetric Operations with Surface Margins. in *Computer Vision and Pattern Recognition: Technical Sketches*, (2001).
37. Wu, M., and R. Balakrishnan, Multi-finger and Whole Hand Gestural Interaction Techniques for Multi-User Tabletop Displays. in *ACM Symposium on User Interface Software and Technology*, (2003), 193-202.
38. Zhang, Z., Y. Wu, Y. Shan, and S. Shafer, Visual Panel: Virtual Mouse, Keyboard and 3D Controller with an Ordinary Piece of Paper. in *Workshop on Perceptual User Interfaces (PUI 2001)*, (2001).

APPENDIX

We present a technique to characterize a flow field as simultaneous rotation, uniform scaling, and two-dimensional translation in the image plane. For the flow field described by $\mathbf{x}_i = [x_i \quad y_i]^T$ and $\mathbf{dx}_i = [dx_i \quad dy_i]^T$, each point $\mathbf{x}_i$ moves to $\mathbf{x}'_i = [x'_i \quad y'_i]^T = \mathbf{x}_i + \mathbf{dx}_i$ by rotation θ in the image plane, uniform scaling s and translation $\mathbf{t}$:

$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{1}$$

$$\mathbf{x}'_i = s\mathbf{R}\mathbf{x}_i + \mathbf{t} \tag{2}$$

We first solve for rotation. With means $\bar{\mathbf{x}} = \frac{1}{N}\sum_i \mathbf{x}_i$ and $\bar{\mathbf{x}}' = \frac{1}{N}\sum_i \mathbf{x}'_i$ we may solve for θ [9]:

$$\tan\theta = \left[\frac{\sum_i (x_i - \bar{x})(y'_i - \bar{y}') - (y_i - \bar{y})(x'_i - \bar{x}')}{\sum_i (x_i - \bar{x})(x'_i - \bar{x}') + (y_i - \bar{y})(y'_i - \bar{y}')} \right] \tag{3}$$

Scaling factor s and translation $\mathbf{t} = [t_x \quad t_y]^T$ may be recovered by least squares:

$$\mathbf{z} = [s \quad t_x \quad t_y]^T \tag{4}$$

$$\mathbf{M}_i = \begin{bmatrix} \mathbf{R}\mathbf{x}_i & \begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix} \end{bmatrix} \tag{5}$$

$$\mathbf{x}'_i = \mathbf{M}_i \mathbf{z} \tag{6}$$

$$\mathbf{z} = \left(\sum_i \mathbf{x}'^T_i \mathbf{M}_i \right) \left(\sum_i \mathbf{M}_i{}^T \mathbf{M}_i \right)^{-1} \tag{7}$$

It may not be obvious that this formulation allows for rotation and scaling about any point. For example, consider rotation about a point $\mathbf{t}_R$:

$$\begin{aligned} \mathbf{x}'_i &= s(\mathbf{R}(\mathbf{x}_i - \mathbf{t}_R) + \mathbf{t}_R) + \mathbf{t} \\ &= s\mathbf{R}\mathbf{x}_i + \mathbf{t}' \end{aligned} \tag{11}$$

where with $\mathbf{t}' = -s\mathbf{R}\mathbf{t}_R + s\mathbf{t}_R + \mathbf{t}$ we arrive at the original form of equation 2.