Spatial Control of Interactive Surfaces in an Augmented Environment

Stanislaw Borkowski, Julien Letessier, and James L. Crowley

Project PRIMA, Lab. GRAVIR-IMAG
INRIA Rhône-Alpes, 655, ave de l'Europe
38330 Montbonnot, France
{Stan.Borkowski, Julien.Letessier, James.Crowley}@inrialpes.fr

Abstract. New display technologies will enable designers to use every surface as a support for interaction with information technology. In this article, we describe techniques and tools for enabling efficient man-machine interaction in computer augmented multi-surface environments. We focus on explicit interaction, in which the user decides when and where to interact with the system. We present three interaction techniques using simple actuators: fingers, a laser pointer, and a rectangular piece of cardboard. We describe a graphical control interface constructed from an automatically generated and maintained environment model. We implement both the automatic model acquisition and the interaction techniques using a Steerable Camera-Projector (SCP) system.

1 Introduction

Surfaces dominate the physical world. Every object is confined in space by its surface. Surfaces are pervasive and play a predominant role in human perception of the environment. We believe that augmenting surfaces with information technology will act as an interaction modality easily adopted for a variety of tasks. In this article, we make a step towards making this a reality.

Current display technologies are based on planar surfaces [8, 17, 23]. Displays are usually treated as access points to a common information space, where users manipulate vast amounts of information with a common set of controls. Given recent developments in low-cost display technologies, the available interaction surface will continue to grow, forcing the migration of interfaces from a single, centralized screen to many, space-distributed interactive surfaces. New interaction tools that accommodate multiple distributed interaction surfaces will be required.

In this article, we address the problem of spatial control of an interactive display surface within an office or similar environment. In our approach, the user can choose any planar surface as a physical support for interaction. We use a steerable assembly composed of a camera and video projector to augment surfaces with interactive capabilities. We exploit our projection-based augmentation to attain three goals: *(a)* mod-

elling the geometry of the environment by using it as a source of information, *(b)* creation of interactive surfaces anywhere in the scene, and *(c)* realisation of novel interaction techniques through augmentation of a handheld display surface.

In the following sections, we present the technical infrastructure for experimentation with multiple interactive surfaces in an office environment (Sections 3 and 4). We then discuss spatial control of application interfaces in Section 5. In Sections 6, 7 and 8 we describe three applications that enable explicit control of interface location. We illustrate interaction techniques with a single interaction surface controlled in a multi-surface environment, but we emphasize that they can be easily extended to the control of multiple independent interfaces controlled within a common space.

2 Camera-Projector Systems

Camera-projector systems are increasingly used in augmented environment systems [11, 13, 21]. Projecting images is a simple way of augmenting everyday objects and allows alteration of their appearance or function. Associating a video projector with a video camera offers an inexpensive means of making projected images interactive. However, standard video-projectors have small projection area which limits their flexibility in creating interaction spaces. We can achieve some steerability on a rigidly mounted projector by moving sub windows within the cone of projection [22], but extending or moving the display surface requires increasing the angle range of the projector beam. This requires adding more projectors, an expensive endeavor. An alternative is to use a steerable projector [2, 12]. This approach is becoming more attractive, due to a trend towards increasingly small and inexpensive video projectors.

Projection is an ecological (non-intrusive) way of augmenting the environment. Projection does not change the augmented object itself, only its appearance. Augmentation can be used to supplement the functionality of objects. In [12], ordinary artefacts such as walls, shelves, and cups are transformed into informative surfaces, but the original functionality of the objects does not change. The objects become physical supports for virtual functionalities. An example of object enhancement is presented in [1], where users can interact with both physical and virtual ink on a projection-augmented whiteboard.

While vision and projection-based interfaces meet most of the ergonomic requirements of HCI, they suffer from lack of robustness due to clutter and insufficiently developed methods for text input. People naturally avoid obstructing projected images, so occlusion is not a problem when camera and projector share the same viewpoint. As for the issue of text input on projected steerable interfaces, currently available projected keyboards like the Canesta Projection Keyboard [16] rely on hardware configuration, which excludes their use on arbitrary surfaces. Resolving this issue is important for development of projection-based interfaces, but it is outside the scope of this work.

3 The Steerable Camera-Projector System

In our experiments, we use a Steerable Projector-Camera (SCP) assembly (Figure 1). It enables us to experiment with multiple interactive surfaces in an office environment.

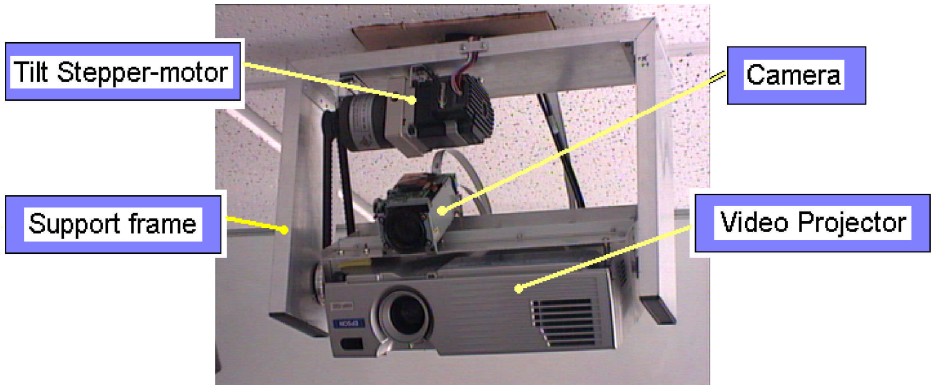

Fig. 1. The Steerable Camera-Projector pair.

The Steerable Camera-Projector (SCP) platform is a device that gives a video-projector and its associated camera two mechanical degrees of freedom, pan and tilt. Note that the projector-camera pair is mounted in such a way that the projected beam overlaps with the camera view. Association of the camera and projector creates a powerful actuator-sensor pair enabling observation of users' actions within the camera field of view. Endowed with the ability to modify the scene using projected light, projector-camera systems can be exploited as sensors (Section 5.2).

4 Experimental Laboratory Environment

The experiments described below are performed in our Augmented Meeting Environment (AME). The AME is an ordinary office equipped with ability to sense and act. The sensing infrastructure includes five steerable cameras, a fixed wide angle camera, and a microphone array. The wide angle camera has a field of view that covers the entire room. Steerable cameras are installed in each of the four corners of the room. A fifth steerable camera is centrally mounted in the room as part of the steerable camera-projector system (SCP).

Within the AME, we can define several surfaces suitable for supporting projected interfaces. Some of these are marked by white boundaries in Figure 2. These regions were detected by the SCP during an automatic off-line environmental model building phase described below (Section 5.2). Surfaces marked with dashed boundaries can be optionally calibrated and included in the generated environment model using the device described in Section 8.

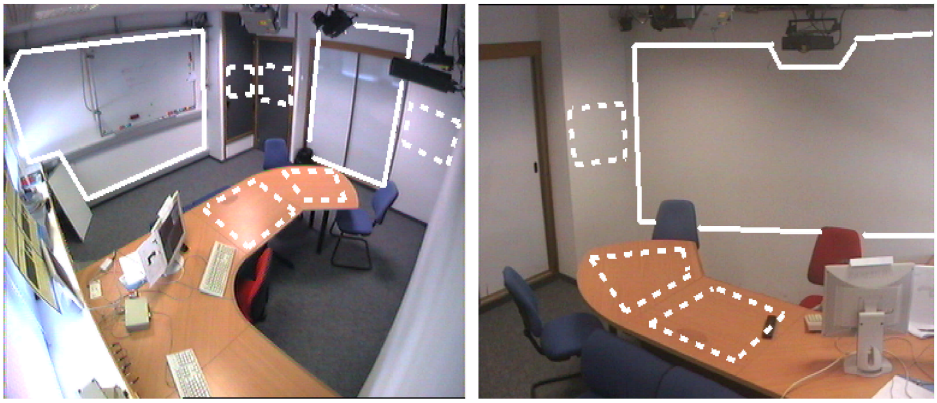

Fig. 2. Planar surfaces in the environment.

5 Spatial control of displays

Interaction combines action and perception. In an environment where users may interact with a multitude of services and input/output (IO) devices, both perception and interaction can be complex. We present a sample scenario in Section 5.1 and describe our approach to automatic environment model acquisition in Section 5.2, but first we discuss the relative merits of our approach to interaction within an augmented environment.

Explicit vs. Implicit. Over the last few years, several research groups have experimented with environments augmented with multiple display surfaces using various devices such as flat screens, whiteboards, video-projectors and steerable video-projectors [3, 8, 11, 13, 21, 23]. Most of these groups focuse on the integration of technical infrastructure into a coherent automated system, treating the problem of new methods for spatial control of interfaces as a secondary issue. Typically, the classic paradigm of drag and drop is used to manipulate application interfaces on a set of wall displays and table display [8]. In such systems, discontinuities in the transition between displays disrupt interaction and make direct adaptation of drag and drop difficult.

An alternative is to liberate the user by letting the system take control of interface location. In [11], the steerable display is automatically redirected to the surface most appropriate for the user. Assuming a sufficient environment model, the interface follows the user by jumping from one surface to another. However, this solution has disadvantages. For one, it requires continuous update of the environment model. More importantly, the system has to infer if the user wants to be followed or not. Such a degree of understanding of human activity is beyond the state of the art.

The authors in [3] combine automatic and explicit control. By default, the interface follows its owner in the augmented room. The user can also choose a display from a list. However, their approach assumes that the user is able to correctly identify the listed devices. Moreover, the method of passing back and forth from automatic to manual control mode is not clearly defined. In this work, we focus on developing interaction techniques that enable users to explicitly control the interface position in space.

Ecological vs. Emmbedded. In ubiquitous computing, panoply of small interconnected devices embedded in the environment or worn by the user are assumed to facilitate continuous and intuitive access to virtual information spaces and services. Many researchers follow this approach and investigate new interaction types based on sensors embedded in artifacts or worn by users [14, 18, 19]. Although embedding electronic devices leads to a number of efficient interface designs, in many circumstances it is unwise to assume that everyone will be equipped with the necessary technology. Moreover, as shown in [1, 3], one can obtain pervasive interfaces by embedding computational infrastructure in the environment instead. Our approach is to create new interaction modes and devices by augmenting the functionality of mundane artifacts without modifying their primary structure.

User-centric vs. Sensor-centric. Coutaz *et al.* [7] highlight the duality of interactive systems. We apply this duality to the analysis of environment models, extending our understanding of the perceived physical space. When building an environment model, the system typically generates a sensor-centric representation of the scene, but this abstraction is not necessarily comprehensible for the human actor. A common understanding of the environment requires translation of the model into a user-centric representation. Such an approach is presented in [3], where the authors introduce an interface for controlling lights in a room. Lamps are shown graphically on a 2D map of the environment, and the user chooses from the map which light to dim or to brighten. The problem is that modeling the real-world environment in order to generate and maintain a human-comprehensible representation of the space is a difficult and expensive task. Moreover, from the user's perspective, the physical location of the controlled devices is not as important as the effect of changing a device's state. Rather than showing the user a symbolic representation of the world, we enrich the sensor-centric model with contextual cues that facilitate mapping from an abstract model to the physical environment.

In summary, we impose the following constraints on multi-surface systems:

1. Users have control of the spatial distribution of applications when they have direct or actuator-mediated access to its interface.
2. Users can control the system both "as they come" without specific tools, and with the use of control devices.
3. The mapping between the symbolic representation of the controller interface and the real world is understandable by an unexperienced user, provided sufficient contextual cues.
4. The underlying sensor-centric model of the environment is generated and updated automatically.

In the following section, we illustrate our expectations of a multi-surface interaction system with a scenario.

5.1 Scenario

John, a professor in a research laboratory, is in his office preparing slides for a project meeting. As the project partners arrive, John hurryly moves the presentation he just finished to a large wall-mounted screen in the meeting room, choosing it from a list of available displays. The list contains almost twenty possible locations in his office and in the meeting room. John has no trouble making his selection because the name of each surface is beside its image as it appears in the scene.

During the meeting, John uses a wide screen to present slides about software architecture. John uses an ordinary laser-pointer to highlight important elements in the slide. The slides are also projected onto a whiteboard so that John can make notes directly on them by drawing on the white board with an ink pen. On command he can record his notations in a new slide that combines his notations with the projected material. At one point, John sees that there is not enough free space on the white board, so he decides to move the projected slide to free some space for notes. He "double-blinks" the laser-pointer on the image, so that the image follows the laser dot.

While the project participants discuss the problem at hand, it becomes apparent that it is useful to split the meeting in three sub-workgroups. John takes one of the groups to his office. From the display list, John chooses the largest surface in his office. He sends the slide to this surface. A second group gathers around the desk in the meeting room. John sends the relevant slide from the wide screen to the desk with the use of a laser-pointer. The third smaller group decides to work in the back of the meeting room. Since there is no display, they take a cardboard onto which they transfer their application interface. They continue their work by interacting directly with the interface projected on the portable screen.

5.2 Environment modeling and image rectification

In our approach to human-computer interaction, it is critical that the system is aware of its working space in order to provide appropriate feedback to the user. The graphical user interfaces enabling explicit control of the display location (Sections 6 and 7) are generated based on the environment model. They contain information facilitating mapping of the virtual sensor-centric model to the physical space.

Although 3D environment models have many advantages for applications involving the use of steerable interfaces, they are difficult to create and maintain. One often makes the simplifying assumption that they exist beforehand and do not change over time [3, 11]. Instead, we propose automatic acquisition of a 2D environment model. The model consists of two layers: *(a)* a labelled 2D map of the environment in the SCP's spherical coordinate system and *(b)* a database containing the acquired characteristics for each detected planar surface. Our environment model directly reflects the available sensor capabilities of our AME.

To acquire the model of the environment, we exploit the SCP's ability to modify the environment by projecting and controlling images in the scene. Model acquisition consists of two phases: first, planar surfaces are detected and labelled with unique identifiers, and second, an image of each planar surface is captured and stored in the model database. In the second phase, the system projects a sample image on each planar surface detected in the environment model and takes a shot of the scene with the camera that has the projected image in its field of view. The images show the available interaction surfaces together with their surroundings. They are used later-on to provide users with contextual information which facilitates the mapping between the sensor-centric environment model and the physical world.

In order to customize the system, users should have the ability to supplement or replace the images in the model database with other data structures (e.g. text labels or video sequences). Using an interaction tool described in Section 8, the model is updated each time a new planar surface is defined in the environment.

Detection of planar surfaces. Most existing methods for projector-screen geometry acquisition provide a 3D model of the screen [5, 25]. However, such methods require the use of a calibrated projector-camera pair separated by a significant base distance. Thus, they are not suitable for our laboratory. In our system, we employ a variation of the method described in [2]. We use a steerable projector and a distant non-calibrated video camera to detect and estimate orientation of planar surfaces in the scene.The orientation of a surface with respect to the beamer is used to calculate a pre-warp that is applied to the projected image. The pre-warp compensates for oblique projective deformations caused by the non-orthogonality of the projector's optical axis relative to the screen surface. Note that the pre-warped image uses only a subset of the available pixels. When images are projected at extreme angles, the effective resolution can drop to a fraction of the projector's nominal resolution. This implies the need for an interface layout adaptation mechanism, that takes into account readability of the interface at a given projector-screen configuration. Adaptation of interfaces is a vast research problem and is not treated in this work.

6 Listing the available resources

In this section, we present a menu-like automatically generated interface enabling a user to choose the location of the display or application interface.

Pop-up and scroll-down menus are known in desktop-based interfaces for at least twenty years. Since planar surfaces in the environment can be seen as potential resources, it is natural to use a menu as a means for choosing a location for the interface.

Together with the projected image as application interface, we project an interactive button that is sensitive to touch-like movements of the user's fingertip. When the user touches the button, a list of available screen locations appears (Figure 3).

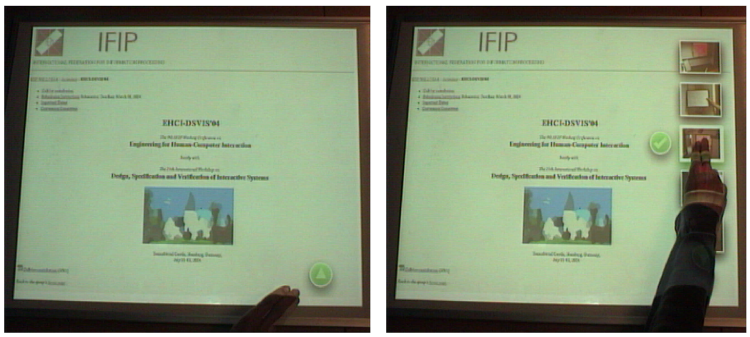

Fig. 3. Interacting with a list of displays (envisionment).

As mentioned in Section 5, we enhance the controller interface with cues that help map the interface elements to the physical world. Therefore, we present each list item as an image taken by one of the cameras installed in the room. We automatically generate the list based on images taken during the off-line model building process (Section 5.2). The images show the available interaction surfaces together with their surroundings. The user chooses a new location for the interface by passing a finger over a corresponding image. Note that one of the images shows a white cardboard, which is an interaction tool described in Section 8. In order to avoid accidental selection, we include a "confirm" button. The user cancels the interaction with the controller application by touching the initialization button again. The list also disappears if there is no interaction for a fixed period of time.

One can easily extend our image-based approach for providing contextual cues from interface control to general control of visual-output devices. For example, instead of showing a map of controllable lamps in a room, we can display a series of short sequences showing the corresponding parts of the room under changing light settings. This allows the user to visualize the effects of interaction with the system before actual execution.

6.1 Vision-based touch detection

Using vision as an user-input device for a projected interface is an elegant solution because *(a)* it allows for direct manipulation, i.e. no intermediary pointing device is used, and *(b)* it is ecological – no intrusive user equipment is required, and bare-hand interaction is possible. This approach has been validated by a number of research projects, for instance the DigitalDesk [24], the Magic Table [1] or the Tele-Graffiti application [20].

Existing vision-based interactive systems track the acting member (finger, hand, or head) and produce actions (visual feedback and/or system side effects) based on recognized gestures. One drawback is that a tracking system can only detect apparition, movement and disparition events, but no "action" event comparable to the mouse-click in conventional user interfaces, because a finger tap cannot be detected by a vision system alone [24]. In vision-based UIs, triggering a UI feature (e.g. a

button widget) is usually performed by holding (or "dwelling") the actuator (e.g. over the widget) [1, 20].

Various authors have tried different approaches to finger tracking, such as correlation tracking, model-based contour tracking, foreground segmentation and shape filtering, etc. While many of these are successful in constrained setups, they perform poorly for a projected UI or in unconstrained environments. Furthermore, they are computationally expensive. Since our requirements are limited to detecting fingers dwelling over button-style UI elements, we don't require a full-fledged tracker.

Approach. We implement an appearance-based method based on monitoring the perceived luminance over UI widgets. Consider the two areas depicted in Figure 4.

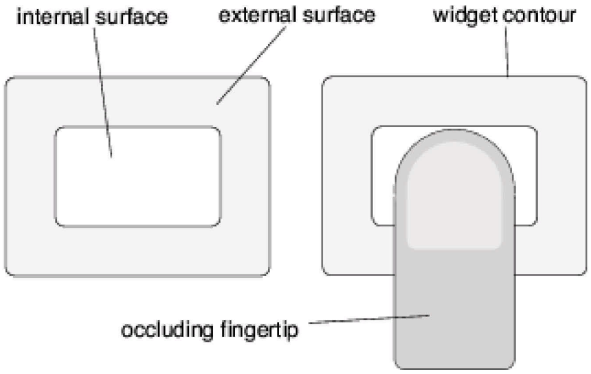

Fig. 4. Surfaces defined to detect touch-like gestures over a widget.

The inner region is assumed to roughly be of the same size as a finger. We denote $L_o(t)$ and $L_i(t)$ to be the average luminance over the outer and inner surface at time t, and

$$\Delta L(t) := \left| L_o(t) - L_i(t) \right|$$

Assuming that the observed widget has a reasonably uniform luminance, ΔL is close to zero at rest, and is high when a finger hovers over the widget. We define the threshold θ to be twice the median value of $\Delta L(t)$ over time when the widget is not occluded. Given the measured values of $\Delta L(t)$, the system generates the event e_0 (or e_1), at each discrete timestep t when $\Delta L(t) < \theta$ (or $\geq \theta$). These events are fed into a simple state machine that generates a *Touch* event after a dwell delay τ (Figure 5).

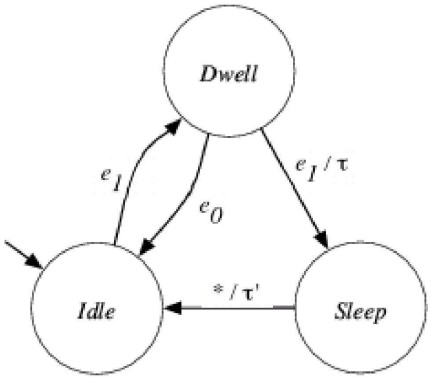

Fig. 5. The finite state machine used to process widget events.

We define two delays: τ to prevent false alarms (the *Dwell* → *Sleep* transition is only triggered after this delay), and τ' to avoid unwanted repetitive triggering (the *Sleep* → *Idle* transition is only triggered after this delay). A *Touch* event is issued whenever entering the *Sleep* state. τ and τ' are chosen equal to 200 ms. This technique achieves robustness against full occlusion of the UI component (e.g. by the user's hand or arm), since such occlusions cause ΔL to remain under the chosen threshold.

Experimental results. Our relatively simple approach provides good results because it is robust to changes in lighting conditions (it is a memory-less process), and occlusions (due to the dynamic nature of event generation and area-based filtering). Furthermore, it is implemented as a real-time process (it runs at camera frequency with less than 50 ms latency), although its cost scales linearly with the number of widgets to monitor.

An example application implemented with our "Sensitive Widgets" approach is shown in Figure 6. The minimal user interface consists of four projected buttons that can be "pressed" i.e. partially occluded with one or more fingers, to navigate through a slideshow.

Using this prototype, we confirm that our approach is robust to arbitrary changes in lighting conditions (the interface remains active during the changes) and full occlusion of widgets.

Integration. We integrate "Sensitive widgets" into a Tk application in an object oriented fashion: they are created and behave as usual Tk widgets. The implementation completely hides the underlying vision process, and provides activation (*Click*) events without uncertainty.

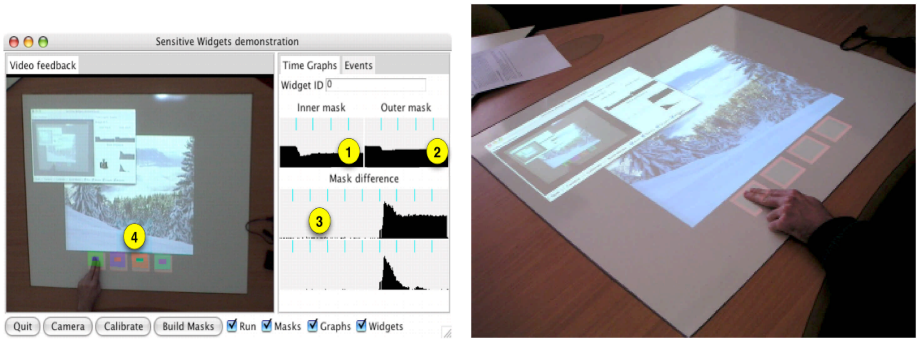

Fig. 6. The "Sensitive Widgets" demonstration interface. *Left:* The graphs exhibit the evolution of a variable in time: *(1)* $L_i(t)$; *(2)* $L_o(t)$; *(3)* $\Delta L(t)$. Notice the high value of ΔL while the user occludes the first widget. The video feedback *(4)* also displays the widget masks as transparent overlays. *Right:* The application interface as seen by the user (the control panel wasn't hidden), in unconstrained lighting conditions (here, natural light).

7 Laser-based control

Having a large display or several display locations demands methods to enable interaction from a distance. Since pointing with a laser is intuitive, many researchers have investigated how to use laser-pointers to interact with computers [4, 9]. Most of them try to translate laser-pointer movements to events similar to those generated by a mouse. According to Myers *et al.* [10], pointing at small objects with a laser is much slower than with standard pointing devices, and less precise compared to physical pointing. On the other hand, pointing with a hand or finger has a very limited range. Standard pointing devices like the mouse or trackball provide interaction techniques that are suitable for a single screen setup, even if the screen is large, but they cannot by adapted to multiple display environments with complex geometry. Hand pointing from a distance provides interesting results [6], but the pointing resolution is too low to be usable, and stereoscopic vision is required.

In our system, we use laser-based interaction exclusively to redirect the beamer (SCP) from one surface to another. This corresponds to moving an application interface to a different location in the scene. Users are free to use their laser pointers in a natural fashion. They can point at anything in the room, including the projected images. The system does not respond unless a user makes an explicit sign.

In our application, interaction is activated with a double sequence of switching the laser on and off while pointing to roughly the same spot on the projected image. If after this sign the laser point appears on the screen and does not move for a short time, the control interface is projected. During the laser point dwell delay we estimate hand jitter in order to scale the controller interface appropriately, as explained below.

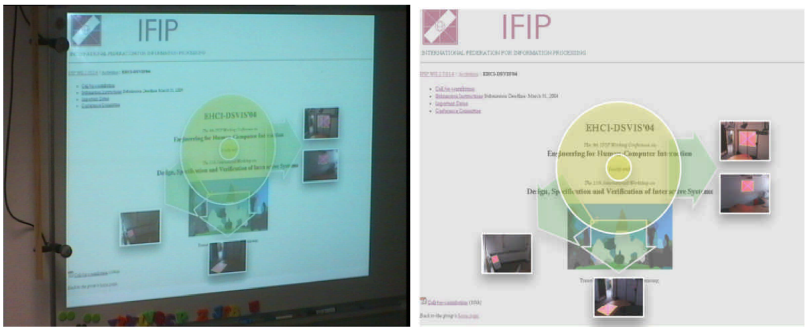

Fig. 7. Laser-based control interface (envisonment)

The interface shown in Figure 7 is a semi-transparent disc with arrows and thumbnail images. The arrows point to physical locations of the available displays in the environment. Similar to the menu-like controller application, the images placed at the end of each arrow are taken from the environment model. They present each display surface as it appears in the scene. The size of the images is a function of the measured laser point jitter. So is the size of the small internal disc representing the dead-zone, in which the laser dot can stay without reaction of the system. The controller interface is semi-transparent in order to avoid breaking users' interaction with the application, in case of a false initialization.

In order to avoid unwanted system reaction, the interface is not active when it appears. To activate it, the user has to explicitly place and keep the laser dot for a short time in any of the GUI's elements (arrow, image or disc). As the user moves the laser point within the yellow outer disc, the system starts to move the interface following the laser point with the center of the disc. This movement is limited to the area of the current display surface. Interface movement is slow for proper user control. When the laser goes outside the yellow disc or enters an arrow, movement halts. The user can then place the laser dot in the image of choice. As the laser point enters an image, the application interface immediately moves across the room to the corresponding surface. The controller interface does not appear on the newly chosen display unless it is again activated. At any time during the interaction process, the user can cancel the interaction by simply switching off the laser pointer.

7.1 Laser tracking with a camera

Several authors have investigated interaction from a distance using a laser pointer [4, 9,10].

Once we achieve geometric calibration of the camera and projector fields of view, detection and tracking the laser pointer dot is a trivial vision problem. Since laser light has a high intensity, a laser spot is the only visible blob on an image captured with a low-gain camera. The detection is then obtained by thresholding the intensity image and determining the barycentre of the connected component. Robustness against false alarms can be achieved by filtering out connected components that have aberrant areas.

As for other tracking systems, the output is a flow of *appear*, *motion* and *disappear* events with corresponding image-space positions. We achieve increased robustness by:

- generating *appear* events only once the dot has been consistently detected over several frames (e.g. 5 frames at 30Hz);
- similarly delaying the generation of *disappear* events.

We are not concerned by varying lighting conditions and shadowing because the camera is set to low gain. Occlusion, on the other hand, is an issue because an object passing through the laser beam causes erratic detections, which should be filtered out.

The overall simplicity of the vision process allows it to be implemented at camera rate (ca. 50Hz) with low latency (ca. 10ms processing time). Thus, it fulfils closed-loop human-computer interaction constraints.

8 A novel user-interface: the PDS

Exploiting robust vision-based tracking of an ordinary cardboard using an SCP unit [2] enables the use of a Portable Display Surface (PDS). We use the SCP to maintain a projected image onto the hand-held screen (PDS), automatically correcting for 3D translations and rotations of the screen.

We extend the concept of the PDS by integrating it in our AME system. As described in the example scenario (Section 5.1), the PDS can be used as a portable physical support for a projected interface. This mode of use is a variation of the “pick and drop” paradigm introduced in [15]. From the system point of view, the only difference between a planar surface in the environment and the PDS is its mobility and the image-correction matrix, so we can project the same interactive-widget-based interface on both static and portable surfaces. In practice, we have to take in account the limits of the image resolution available on the PDS surface.

The portability of this device creates two additional roles for the PDS in the AME system. It can serve as a means for explicit control of the display location and as a tool enabling the user to extend the environment model to surfaces which are not detected during the offline model acquisition procedure. Actually, the two modes are closely coupled and the extension of the environment model is transparent for the user.

To initialize the PDS, the user has to choose the corresponding item in the GUIs described in previous sections. Then, the SCP projects a rectangular region into which the user has to put the cardboard screen. If no rectangular object appears in this region within a fixed delay, the system falls back to its previous state. When the PDS is detected in the projected initialization region, the system transfers the display to the PDS and starts the tracking algorithm. The user can then move in the environment with the interface projected on the PDS. To stop the tracking algorithm, the user touches the “Freeze” widget projected on the PDS. The location of the PDS together with the corresponding pre-warp matrix is thus added to the environment model as new screen surface. This mechanism allows the system to dynamically update the model.

9 Conclusions

The emergence of spatially low-constrained working environments calls for new interaction concepts. This paper illustrates the issue of spatial control of a display in a multiple interactive-surface environment. We use steerable camera-projector assembly to display an interface and to move it in the scene. The projector-camera pair is also used as an actuator-sensor system enabling automatic construction of a sensor-centric environment model. We present three applications enabling convenient control of the display location in the environment. The applications are based on interactions using simple actuators: fingers, a laser pointer and a hand-held cardboard.

We impose a strong relation between the controller application interface and the physical world. The graphical interfaces are derived from the environment model, allowing the user to map the interface elements to the corresponding real-world objects. Our next development step is to couple controller applications with standard operating systems infrastructure.

Acknowledgments

This work has been partially funded by the European project FAME (IST-2000-28323), the FGnet working group (IST-2000-26434), and the RNTL/Proact ContAct project.

References

1. F. Bérard. The magic table: Computer-vision based augmentation of a whiteboard for creative meetings. In *Proceedings of the ICCV Workshop on Projector-Camera Systems*. IEEE Computer Society Press, 2003.
2. S. Borkowski, O. Riff, and J. L. Crowley. Projecting rectified images in an augmented environment. In *Proceedings of the ICCV Workshop on Projector-Camera Systems*. IEEE Computer Society Press, 2003.
3. B. Brumitt, B. Meyers, J. Krumm, A. Kern, and S. Shafer. Easyliving: Technologies for intelligent environments. In *Proceedings of Handheld and Ubiquitous Computing*, September 2000.
4. J. Davis and X. Chen. Lumipoint: Multi-user laser-based interaction on large tiled displays. *Displays*, 23(5), 2002.
5. R. Raskar et al. iLamps: Geometrically aware and self-configuring projectors. In *Appears ACM SIGGRAPH 2003 Conference Proceedings*.
6. Yi-Ping Hungy, Yao-Strong Yangz, Yong-Sheng Cheny, Ing-Bor Hsiehz, and Chiou-Shann Fuhz. Free-hand pointer by use of an active stereo vision system. In *Proceedings of the 14th International Conference on Pattern Recognition (ICPR'98)*, volume 2, pages 1244–1246, August 1998.
7. J.Coutaz, C.Lachenal, and S. Dupuy-Chessa. Ontology for multi-surface interaction. In *Proceedings of the ninth International Conference on Human-Computer Interaction (Interact'2003)*, 2003.

8. B. Johanson, G. Hutchins, T. Winograd, and M. Stone. Pointright: Experience with flexible input redirection in interactive workspaces. *Proceedings of UIST-2002*, 2002.
9. D. R. Olsen Jr and T. Nielsen. Laser pointer interaction. In *ACM CHI'2001 Conference Proceedings: Human Factors in Computing Systems. Seattle, WA*, 2001.
10. B. A. Meyers, R. Bhatnagar, J. Nichols, C.H. Peck, D. Kong, R. Miller, and A.C. Long. Interacting at a distance: measuring the performance of laser pointers and other devices. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves*. ACM Press New York, NY, USA, April 2002.
11. G. Pingali, C. Pinhanez, A. Levas, R. Kjeldsen, M. Podlaseck, H. Chen, and N. Sukaviriya. Steerable interfaces for pervasive computing spaces. In *Proceedings of IEEE International Conference on Pervasive Computing and Communications - PerCom'03*, March 2003.
12. C. Pinhanez. The everywhere displays projector: A device to create ubiquitous graphical interfaces. In *Proceedings of Ubiquitous Computing 2001 Conference*, September 2001.
13. R. Raskar, G. Welch, M. Cutts, A. Lake, L. Stesin, and H. Fuchs. The office of the future: A unified approach to image-based modeling and spatially immersive displays. In *Proceedings of the ACM SIGGRAPH'98 Conference*.
14. J. Rekimoto. Multiple-computer user interfaces: "beyond the desktop" direct manipulation environments. In *ACM CHI2000 Video Proceedings*, 2000.
15. J. Rekimoto and M. Saitoh. Augmented surfaces: A spatially continuous workspace for hybrid computing environments. In *Proceedings of CHI'99, pp.378-385*, 1999.
16. Helena Roeber, John Bacus, and Carlo Tomasi. Typing in thin air: the canesta projection keyboard - a new method of interaction with electronic devices. In *CHI '03 extended abstracts on Human factors in computing systems*, pages 712–713. ACM Press, 2003.
17. N. A. Streitz, J. Geißler, T. Holmer, S. Konomi, C. Müller-Tomfelde, W. Reischl, P. Rexroth, P. Seitz, and R. Steinmetz. i-land: An interactive landscape for creativitiy and innovation. *ACM Conference on Human Factors in Computing Systems*, 1999.
18. N. A. Streitz, C. Röcker, Th. Prante, R. Stenzel, and D. van Alphen. Situated interaction with ambient information: Facilitating awareness and communication in ubiquitous work environments. In *Tenth International Conference on Human-Computer Interaction*, June 2003.
19. Zs. Szalavári and M. Gervautz. The personal interaction panel - a two-handed interface for augmented reality. In *Proceedings of EUROGRAPHICS'97, Budapest, Hungary*, September 1997.
20. N. Takao, J. Shi, , and S. Baker. Tele-graffiti: A camera-projector based remote sketching system with hand-based user interface and automatic session summarization. *International Journal of Computer Vision*, 53(2):115–133, July 2003.
21. J. Underkofflerand B. Ullmer and H. Ishii. Emancipated pixels: Real-world graphics in the luminous room. In *Proceedings of ACM SIGGRAPH*, pages 385–392, 1999.
22. F. Vernier, N. Lesh, and C. Shen. Visualization techniques for circular tabletop interfaces. In *Advanced Visual Interfaces*, 2002.
23. S.A. Voida, E.D. Mynatt, B. MacIntyre, and G. Corso. Integrating virtual and physical context to support knowledge workers. In *Proceedings of Pervasive Computing Conference*. IEEE Computer Society Press, 2002.
24. P. Wellner. The digitaldesk calculator: Tactile manipulation on a desk top display. In *ACM Symposium on User Interface Software and Technology*, pages 27–33, 1991.
25. R. Yang and G. Welch. Automatic and continuous projector display surface calibration using every-day imagery. In *CECG'01*.