Appears in SPIE Image and Video Communications and Processing (IVCP) Conference, San Jose, CA, vol. 5685, 18-20 January 2005.

A Real-Time 3D Interface Using Uncalibrated Cameras

Nelson L. Chang*
Imaging Technology Department, Hewlett-Packard Laboratories
1501 Page Mill Road MS 1203, Palo Alto, CA 94304 USA

ABSTRACT

This paper proposes a real-time 3D user interface using multiple possibly uncalibrated cameras. It tracks the user's pointer in real-time and solves point correspondences across all the cameras. These correspondences form spatio-temporal "traces" that serve as a medium for sketching in a true 3-D space. Alternatively, they may be interpreted as gestures or control information to elicit some particular action(s). Through view synthesis techniques, the system enables the user to change and seemingly manipulate the viewpoint of the virtual scene even in the absence of camera calibration. It also serves as a flexible, intuitive, and portable mixed-reality display system. The proposed system has numerous implications in interaction and design, especially as a general interface for creating and manipulating various forms of 3-D media.

Keywords: mixed-reality interface, 3D design and interaction, uncalibrated cameras, real-time imaging, view synthesis, augmented reality, spatio-temporal tracking

1. INTRODUCTION

As 3-D display technologies improve, 3-D interfaces will play an increasingly important role in the areas of design and interaction, where traditional interaction modes (e.g. keyboard, mouse) may not be as appropriate or intuitive. Product designers and engineers could more naturally design and visualize future products with the appropriate 3-D interfaces. Teachers and students could both benefit from rich 3-D annotations to enhance distance learning and education applications. Consumers, and in particular gamers, might find 3-D-based interfaces much more appealing for navigating through 3-D environments and manipulating various forms of 3-D media.

There has been a lot of previous work related to 3-D interfaces, primarily camera-based solutions that offer practical means for capturing user's actions and intentions. Large scale systems[11,13] create life-sized collaborative interactive spaces through a collection of synchronized cameras and projectors. Their interface often employs sophisticated human identification and tracking, requiring a huge amount of investment and infrastructure. On the other hand, interactive workspaces and augmented reality systems[15,12] are much smaller scale 3-D interfaces that provide the user with a blended mixed-reality view that merges both real-world scenes and virtual 3-D media. These systems require specially calibrated mirrors to achieve the proper effect. Furthermore, all of these interfaces need precise camera calibration and work strictly in the 3-D domain, thereby limiting their portability and the range of applications.

To address these issues, this paper proposes a flexible camera-based user interface called ArcSpace. ArcSpace consists of two or more cameras and a display connected to the same computer. The cameras are oriented toward a common 3-D volume to form an interaction space for the user. The user interacts with the system through use of a pointer. ArcSpace uses real-time image processing to track the pointer's movement in the interaction space and solve point correspondences across all the cameras. These correspondences form spatio-temporal "traces" that serve as a medium for sketching or designing in a true 3-D space. Alternatively, they may be interpreted as gestures or control information to elicit some particular action(s).

With a calibrated set up, ArcSpace can produce 3-D models simply by triangulating the point correspondences of the traces. However, even without calibrated cameras, ArcSpace still enables interactive rendering and 3-D-like

* nelson.chang@hp.com; phone +1.650.857.5563; http://www.hpl.hp.com/personal/Nelson_Chang

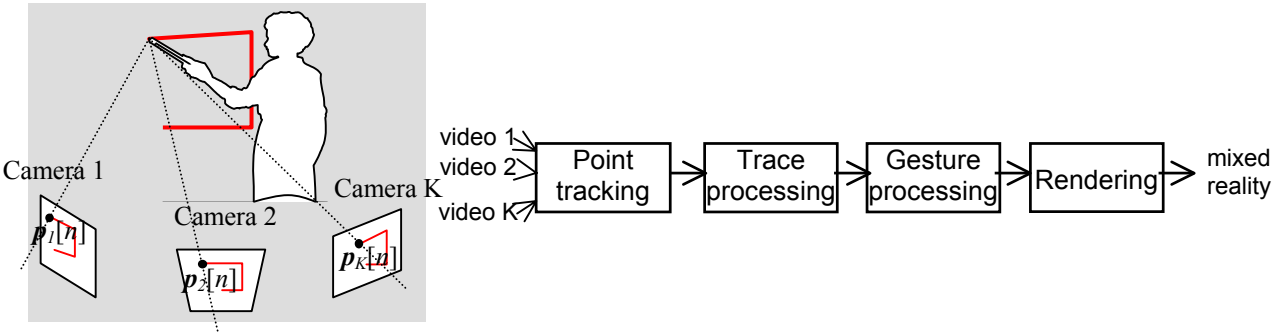

Figure 1. The ArcSpace block diagram.

manipulation of new viewpoints of the traces through view synthesis. It blends virtual traces and user interface components with real-world elements for a mixed-reality visual feedback. In this manner, the system becomes more mobile and flexible for many applications. Moreover, its simplicity enables this 3-D interface to be used on a desktop or as a possible portable solution.

The paper focuses on uncalibrated imaging and is organized as follows. Section 2 describes the four major processing components in ArcSpace. Section 3 provides some experimental results on real-world images from the prototype system. Section 4 discusses extensions to ArcSpace to improve contextual and visual cues for desktop augmented reality. Finally, Section 5 discusses the advantages of the proposed system and directions for future work.

2. ARCSPACE FOR 3-D INTERACTION AND DESIGN

The ArcSpace system captures the user's actions in a 3-D space from a number of cameras and deduces his/her intentions based on predetermined rules and states. In the following, only a single point of interest based on the user's pointer is captured in every frame; if necessary, the approach extends to handle multiple such interest points. As shown in Figure 1, the entire block diagram for the proposed ArcSpace system consists of four major components, each of which is described in greater detail. In the following subsections, $\boldsymbol{p}_i[n]=(u_i[n],v_i[n])$ denotes the 2-D coordinates in Camera i, $i=1,\ldots,K$, at discrete time n, and $c[\boldsymbol{p}_i[n]]=(r(\boldsymbol{p}_i[n]), g(\boldsymbol{p}_i[n]), b(\boldsymbol{p}_i[n]))$ represents its color.

2.1. Real-time point extraction and tracking

The first component identifies and tracks the pointer's coordinates in each of the cameras by foreground extraction. Using the live, synchronized images from the cameras, ArcSpace initially builds up statistics on the static background at every camera pixel $\boldsymbol{p}_i[n]$ by computing the mean color based on the S most recent samples, i.e.

$$\overline{c}[\mathbf{p}_i[n]] = \frac{1}{S}\sum_{m=n-S}^{n-1} c[\mathbf{p}_i[m]] \quad \forall i = 1,\ldots,K\,.$$

The standard deviation $\sigma[\boldsymbol{p}_i[n]]$ is similarly computed at every pixel. To achieve real-time processing with multiple cameras, one can speed up the computation by directly using raw Bayer mosaiced camera images (8 bits per pixel) instead of post-processed 24-bit color images. Furthermore, the $S=4$ most recent samples are sufficient for reasonable performance. Keeping the four most recent 8-bit samples enables an efficient byte-wise shift-and-add approach to quickly update the mean and standard deviation at every camera pixel.

With the background statistics computed, candidate foreground points are identified as points $\boldsymbol{p}_i[n]$ in each camera i that differ significantly from the background colors, i.e. these points are retained if

$$\left\| c[\boldsymbol{p}_i[n]] - \overline{c}[\boldsymbol{p}_i[n]] \right\| > \sigma[\boldsymbol{p}_i[n]] \quad \forall i = 1,\ldots,K\,,$$

where $\sigma[\boldsymbol{p}_i[n]]$ is the standard deviation of the background colors at this coordinate. In this way, statistically similar points (i.e. non-foreground points) are automatically pruned. The candidate foreground points form a binary mask that is morphologically dilated to reduce the effects of noise. This technique efficiently extracts out the foreground object in

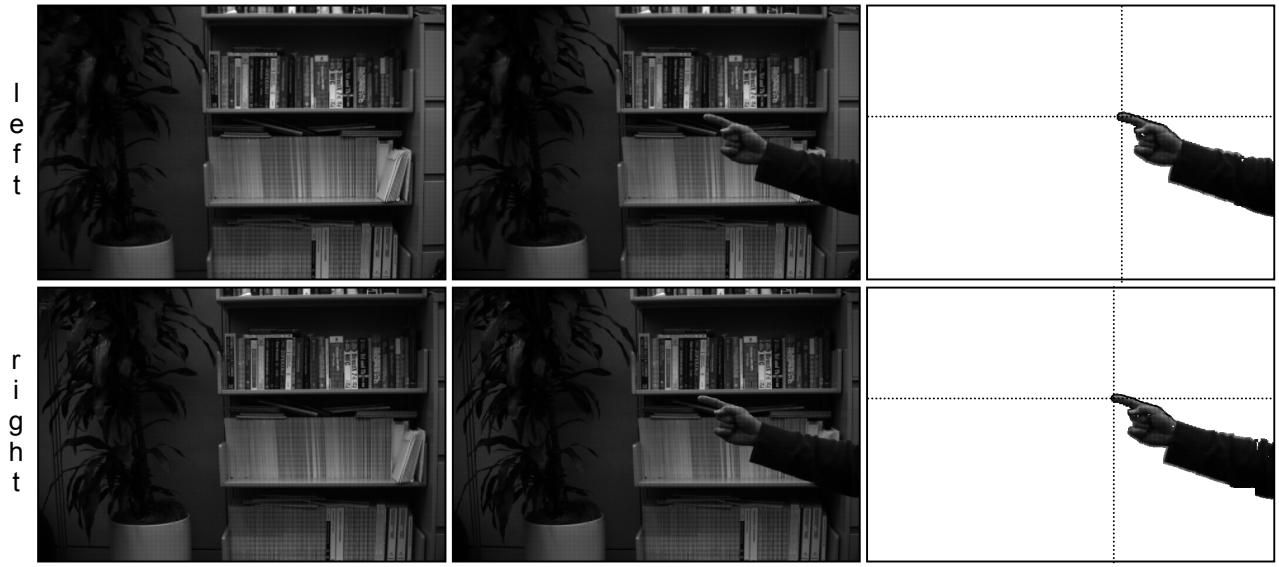

Figure 2. Example of real-time foreground extraction with respect to each camera's reference image.

every camera image. In addition, the background statistics of every non-foreground point is updated to account for lighting and other scene variations.

The point of interest is then computed based on the remaining candidate points in each camera image. When using a pen flashlight or LED as the pointer, one can simply perform pixel thresholding of the M_i candidates $\boldsymbol{q}_{ij}[n]$ in every camera to identify highly saturated points corresponding to the bright pointer. The pointer's location is pinpointed through clustering the largest connected set of points and computing their centroid in every camera, i.e.

$$\boldsymbol{p}_i[n] = \frac{1}{M_i}\sum_{j=1}^{M_i}\boldsymbol{q}_{ij}[n] \quad \forall i = 1,\ldots,K\ .$$

When using a finger or hand as the pointer, one can alternatively apply a breadth-first search on the candidate points in every camera to find the furthest point from the camera's borders; the furthestmost point serves as a reasonably good indicator of the user's intentions.[8] Figure 2 shows a two-camera example of the calculated mean image (raw 8bpp images), the current image, and the extracted object (the user's hand and forearm) in both cameras. The point of interest in each image is automatically found to be the point indicated by the intersecting lines.

In either case, the proposed technique identifies the pointer's location in every camera and thus trivially obtains its correspondence across all the cameras. This observation obviates having to perform highly computational and often error-prone matching algorithms to find the optimal point correspondence. The technique performs reasonably well and is much simpler as compared to approaches that explicitly track human body parts and estimate pose.[1,3]

2.2. Trace processing and construction

The pointer's location in the interaction space determines a set of 2-D coordinates in each camera at every time instance, forming a so-called trace. Over time period (n_{start} to n_{end}), a trace is made up of a series of connected spatio-temporal points in the $2K$+1 dimensional space defined by the K cameras, i.e. $\left\{\boldsymbol{p}_1[n], \boldsymbol{p}_2[n], \ldots, \boldsymbol{p}_K[n]; n\right\}\Big|_{n=n_{start}}^{n_{end}}$. The virtual scene thus collectively consists of a number of possibly disparate traces. These traces may be further organized according to spatial and/or temporal relationships into different objects or design entities.

While occlusions are handled by this framework, only points visible in all cameras are considered when forming traces. Points from successive time instances are added to the same trace until an appropriate state change occurs (as defined in the next subsection). Color information may also be recorded for each trace; in this case, the color remains constant

over all points and during the entire duration. Of course, one can instead capture varying color information for each pixel as a function of time as needed for view- and time-dependent traces.

Multiple 2-D bounding boxes are also computed along with the traces to speed up and improve querying the traces. The bounding boxes are represented by the upper-left and lower-right coordinates in each of the camera spaces. Moreover, they are hierarchically organized: First, a multi-dimensional bounding box is defined for the entire scene in every camera. Next, multi-dimensional bounding boxes are computed for each individual trace. Finally, bounding boxes may be determined for successive line segments in each trace, although the coordinates are precisely the points along the curve itself.

If the relative 3-D geometry of the cameras is established, either through a separate stage involving calibration pattern(s) or even using the traces themselves for self-calibration[2,5], one can greatly simplify the trace data structure. With a calibrated set up, the pointer's 3-D coordinates may be computed directly from the 2-D correspondences through triangulation. In this case, the traces can be reduced from a $2K+1$ quantity to a 4-D quantity (3-D position and time), and the bounding boxes are simply a 3-D bounding volume for the scene and the traces.

2.3. Gesture interpreter and processing

The third processing component focuses on correctly interpreting the inputted traces and executing the appropriate actions. The space of allowable user actions is defined by a predetermined state machine where a given trace may have different meanings based on the system's current state. The primary gesture recognized by ArcSpace is the "hold" gesture, in which the user keeps the pointer in approximately the same location for a period of time (e.g. over one second), i.e.

$$\sum_{m=n-T}^{n-1} \left(\sum_{i=1}^{K} \left\| \boldsymbol{p}_i[n] - \boldsymbol{p}_i[m] \right\|^2 \right) \leq \varepsilon ,$$

where T is at least the frame rate of the system and ε is predetermined bounding threshold. Other gestures (e.g. simple shapes, handwriting) may also be implemented in this system to augment the user's experience.

The prototype ArcSpace system has five main operating modes geared for an example 3-D sketching application: trace drawing, trace moving, trace selection, trace deletion, and hot spot interface. Each of these modes interpret traces and the hold gesture differently.

- In the drawing mode, the user can manipulate the pointer to sketch in the multi-dimensional space using the active color. The hold gesture toggles between drawing states so that the user can still follow the pointer's location without adding to the virtual scene. The gesture also signifies the transition between traces; the previous trace is completed and additional points are added to a new trace.
- In the moving mode, the user can move one or more of the drawn traces, enabling the user to change the current viewpoint. A hold gesture marks the reference point, and its relative distance to the subsequent trace determines the direction and magnitude of the desired movement. For example, with two uncalibrated cameras, the viewpoint is altered by view interpolation, and a lateral pointer movement updates the single view interpolation parameter accordingly.
- In the selection mode, the pointer's location is used as a query to find the nearest trace. The pointer's location is quickly compared with the multi-scale bounding boxes: It is first compared with the scene bounding boxes to determine whether it is within threshold. If so, the pointer's coordinates are subsequently compared with each trace's bounding box. The error distance to the pointer is computed for all traces that are a minimum distance away from the pointer. The algorithm returns and automatically highlights the trace which minimizes the error distance. A hold gesture toggles selection of this trace.
- In the deletion mode, the user can remove one or more traces. A hold gesture serves to remove the selected trace(s), the most recently added trace, or even the entire virtual scene.
- With the hot spot interface, the user can activate various hot spots with the pointer, where a hot spot is a predefined region in the interaction space (either $2K$-dimensional space or 3-D space) corresponding to a particular action or event. For example, one can define a set of 2-D rectangles in each camera's view to act as a multi-dimensional button. If the user positions the pointer such that its projection intersects each of these rectangles, the button may become highlighted or change color, providing a visual "hovering" effect. Moreover, if

the user holds the pointer over the hot spot's location for a sufficient period of time, then the corresponding action of the button is executed. Example actions include switching to one of the above operating modes and changing the active ink color from a palette.

2.4. Rendering and view interpolation

The final processing component produces the rendered output and gives the user visual feedback. The traces of the virtual scene are first updated according to the user-specified viewpoint. If the system is calibrated, then it is straightforward to apply 3-D transformations to render the updated scene. However, ArcSpace enables new view synthesis even in the absence of calibration. In this case, the updated scene is recomputed using a fast view interpolation approach.[4] For K cameras, there are K-1 independent view interpolation parameters specified by the user's movement. The new coordinates $\boldsymbol{p}'[n]$ of the virtual data are simply a linear combination of the K coordinates defined with respect to some reference coordinate system (typically the camera view to be displayed) given by

$$\boldsymbol{p}'[n] = \sum_{i=1}^{K} w_i \, \boldsymbol{p}_i[n] \quad s.t. \sum_{i=1}^{K} w_i = 1 \, .$$

Color may be similarly transformed; however, it is not necessary since the color of a given point is exactly the same in all cameras. The view interpolation parameters w_i may be specified through a graphical user interface[5] or else may be derived from the relative motion of the pointer traces.

To render the virtual scene, every point in each trace goes through the above transformation to determine the new coordinates in the reference frame. Neighboring pairs of the transformed points are drawn as connected line segments, splines, or otherwise interpolated. The appropriate color is applied to each of these segments. One can thus obtain the view of the virtual scene rendered at the desired viewpoint without requiring a calibrated set up. The new views give the appropriate motion parallax cues, and the result can be quite dramatic depending on the placement of the cameras.

When blending real and virtual elements in the scene, it is important to preserve correct depth ordering for proper visual cues. One could estimate the depth of every point in the scene; however, it may be quite computational, making it difficult to maintain real-time rates. Instead, it is assumed that the interaction space lies entirely in front of the background scene. Since virtual objects are rendered last (i.e. closest to view), the only conflict occurs when a virtual object is rendered to occlude a real one when in fact the real object should be in front. A simple but effective technique compares the relative disparities[†] at every virtual point that intersects the real object; if the disparity of the real object is greater, then the real object actually occludes the virtual data at this location and thus the virtual point/segment is not rendered.

The visual feedback to the user is the combination of the virtual data, the user interface (e.g. hot spots), and possibly the live/real video from one or more of the cameras. A simple approach is to first paint the output image with the real video feed from one of the cameras and then superimpose the transformed virtual scene (with the appropriate hidden surfaces removed) and user interface. One may also apply view interpolation to the user interface elements and the pointer's location before blending with the output image. The transformed pointer is marked in each camera view to give the user context of the pointer's location in the interaction space.

As described, the final result may be displayed on a traditional 2-D display. To further enhance the user experience, one can render the results to a 3-D display. With a stereoscopic display (e.g. polarized/red-cyan/side-by-side stereo pairs), real world and virtual objects may appear in 3-D and naturally rendered at the corrected depth without any additional work. For best results, two of the cameras in the set up are properly positioned to provide the left and right stereo views. One can rectify the images to remap epipolar lines to scanlines and minimize vertical offsets; however, it is sufficient to place the cameras on a stereo mounting bar attached to a single tripod and solve for a global vertical offset (directly from the traces) to still produce a compelling stereo effect.

[†] The disparities can be computed by solving for the fundamental matrix between every camera and the reference camera directly from a sufficient number of input traces and then comparing the distance and direction of the multi-dimensional correspondences to the epipoles. A reasonable approximation and speed-up is to assign the disparity of the interest point to the entire foreground object.

Figure 3. ArcSpace automatically preserves relative depth ordering between real and virtual objects in the scene.

Figure 3 gives the example of a rendered stereo pair for a two-camera set up. The user's finger and forearm lie in between two previously drawn virtual objects (virtual red square and a yellow circle) in the interaction space. The aforementioned technique properly hides the lower-right portion of the yellow circle that is occluded by the user's hand. The green dot indicates the interest point computed based on this foreground object. Moreover, when viewed in stereo, the real and virtual objects of this stereo pair, including the background scene, all appear at the correct relative depths.

3. EXPERIMENTAL RESULTS

The experimental system consists of a single 1.13 GHz Pentium III notebook with two Point Grey Research Dragonfly Firewire VGA cameras (up to four are supported on a single bus). The cameras are automatically synchronized and capture at 30 Hz. They are mounted on a stereo bar, placed arbitrarily next to each other and otherwise uncalibrated, where the left camera serves as the reference view. The background consists of the live camera view of an office setting to give the user visual context; when the virtual annotations are unrelated to the background, one could choose instead to display an arbitrary image or nothing at all. Because of the temporal nature of real-time interaction, it is quite difficult to convey the results of the proposed system in a printed or electronic document. Nonetheless, the following example screenshots are representative and demonstrate the effectiveness of ArcSpace.

Figure 4 shows several screenshots taken from a single interaction session. The top row of the display shows various hot spots that the user can invoke including an eight-color palette picker and buttons to switch between different operating modes The user sketches four objects of different colors and depths (Figures 4 (a)-(c)) using a pen flashlight as a pointer. The user then chooses the "selection mode" hot spot (Figure 4 (c)) and the red rectangular shape is found to be the closest to the pointer's position in the interaction space, automatically highlighted in white (Figure 4 (d)). Next, the user manipulates the virtual scene by moving the selected trace through view interpolation (Figure 4 (e)). The user proceeds to delete the selected trace (Figure 4 (f)) and interactively control the view interpolation parameter for the entire virtual scene (Figures 4 (g) and (h)). Note the upper yellow foreground object is correctly moved a greater distance than the rest of the scene to give the correct parallax cues.

Figure 5 highlights the power of view interpolation with ArcSpace. Even with uncalibrated cameras, the user can interactively alter the view to give the parallax-correct appearance of rotating the background happy face with respect to the foreground green rectangle. In this example, the viewpoint of the live images of the real-world background scene is not updated; these views can also be easily updated and blended in this framework with appropriate depth information.

Figure 6 displays the mixed reality output of a couple examples in the form of stereo pairs. In these cases, the relative vertical offset between the two cameras is estimated merely to improve the stereoscopic display effect. ArcSpace captures virtual annotations (i.e. the hat and arrows) simultaneously in the cameras. When viewed in 3-D, the annotations of the top stereo pair automatically appear at the correct depths relative to the real world scene, providing a

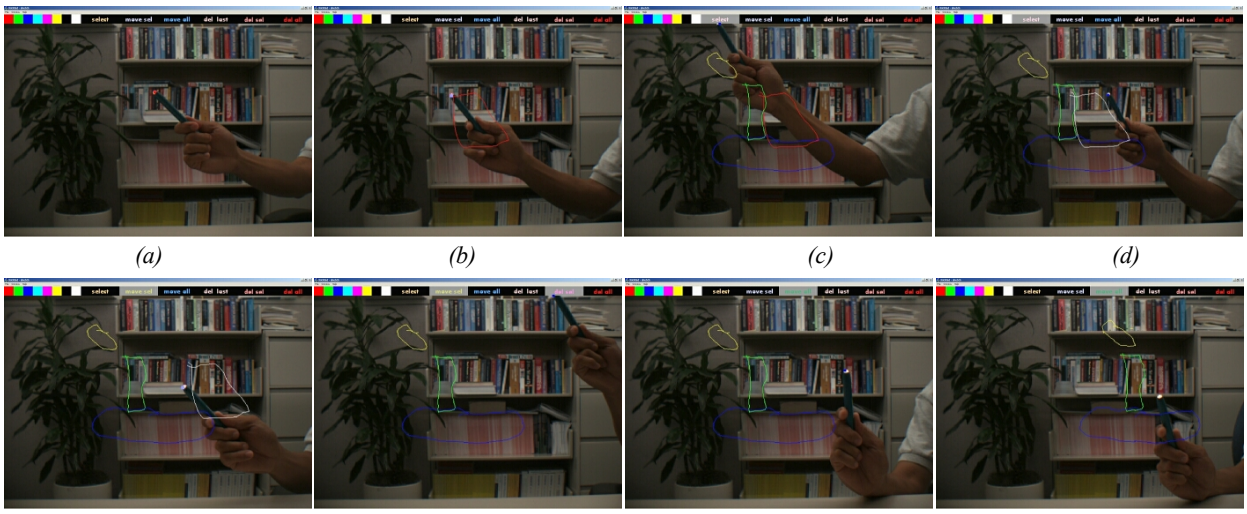

Figure 4. Example screenshots of using the ArcSpace system for drawing, manipulating, and interfacing in a 3-D space.

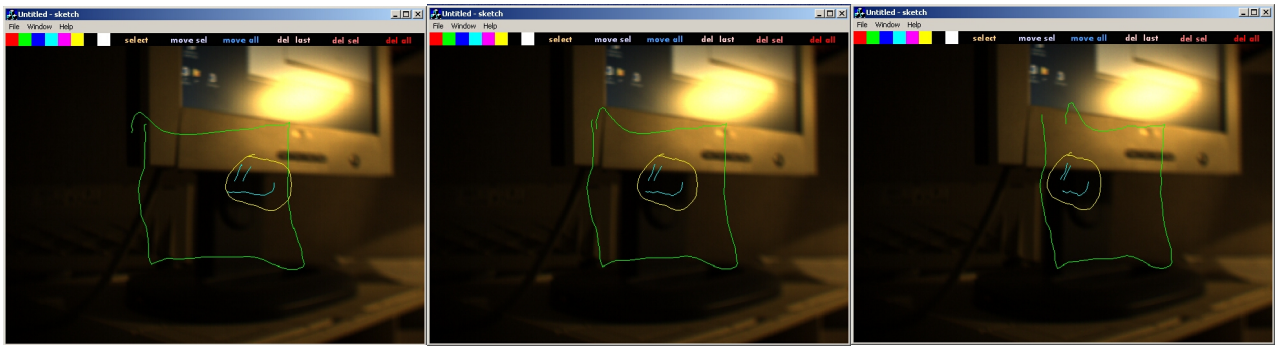

Figure 5. ArcSpace synthesizes parallax-corrected new views even with uncalibrated cameras..

very convincing mixed-reality 3-D result. Specifically, the red left-hand arrow points to a particular leaf, the yellow arrow points to the Buddha mask's left ear, and the green hat appears to be placed correctly with respect to the mask. In the bottom stereo pair, the virtual objects appear to hover at the correct relative location in the 3-D space in front of the user, creating a true sense of dimension.

4. IMPROVING CONTEXTUAL AND VISUAL CUES

For seamless operation, the virtual scene should ideally correlate with the real world perfectly. Typical 2-D-based interfaces employ a planar surface (e.g. presentation screen, table, glass surface) that serves as both the display space and the interaction space.[16,7,6,8,17] Having the common planar surface automatically registers the two spaces, thereby giving the user the appropriate visual cues.

In the case of camera-based 3-D interfaces, however, such an augmented reality experience is much harder to achieve; the user typically performs his/her interactions in one space and looks at a different space to see the updated display.[9] Hence, the user does not get the proper contextual or visual cues for the virtual scene, thereby breaking the immersive mixed-reality experience. For example, with the cameras pointing in front of a monitor, a virtual box drawn in the scene does not physically appear in the interaction space especially when the user's viewpoint changes. Head-mounted displays with head trackers offer one expensive and rather cumbersome solution to overcome this problem. Another solution comes in the form of sophisticated desktop worksurfaces that require mirrors and precise 3-D calibration for proper operation.

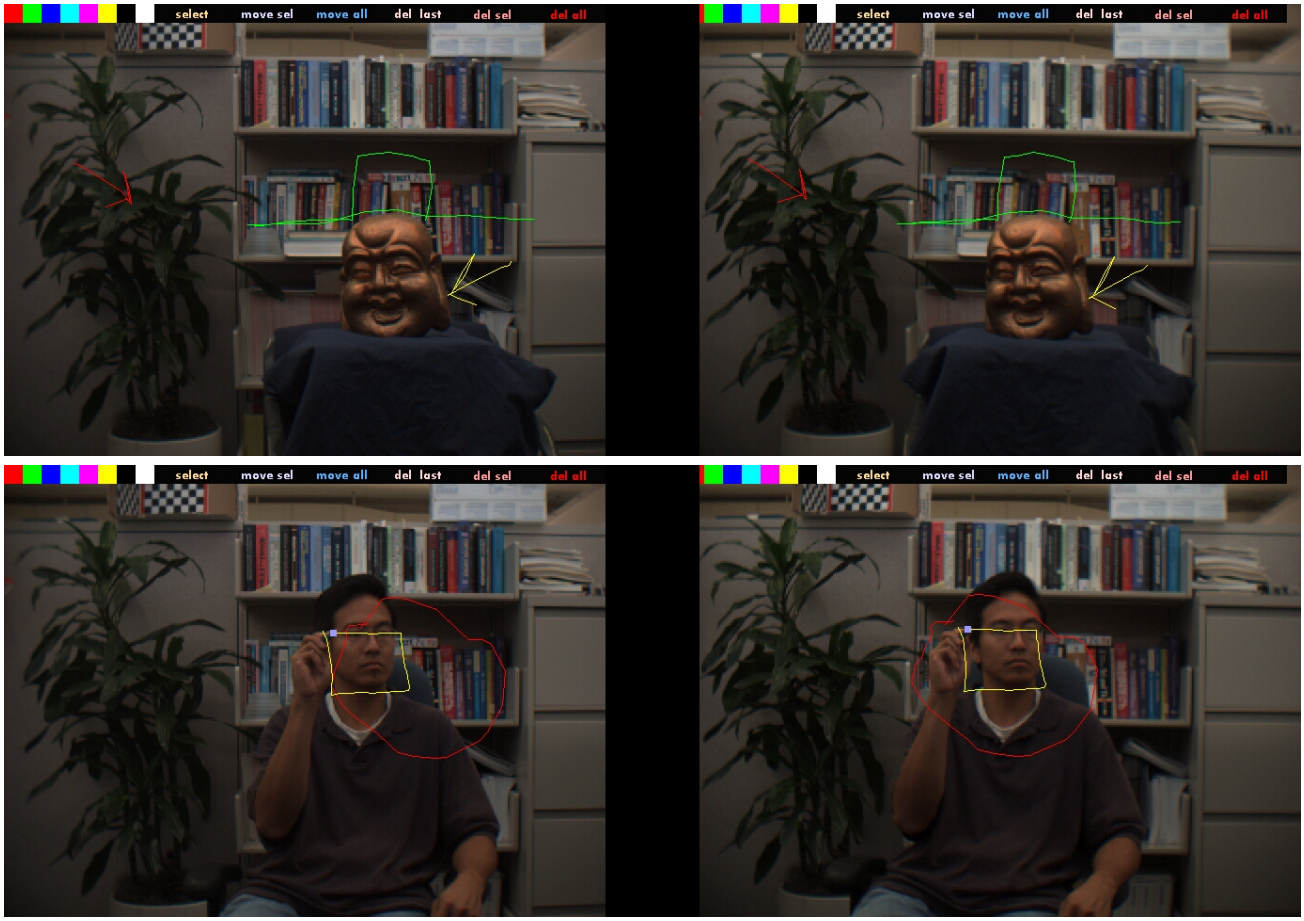

Figure 6. Virtual annotations in captured stereo pairs automatically appear at the correct relative depth in the scene.

A simple solution is to instead use the ArcSpace system with the cameras oriented *behind* the display. As shown in Figure 7, the interaction space then becomes the 3-D volume located behind the display surface. In this "through the looking glass" paradigm, the virtual scene is automatically registered to the real world without the need for explicit calibration. In this case, the display serves as the separating yet common interface between the viewing and interaction spaces. Even with a traditional 2-D display, having multiple cameras behind the display helps to capture the multi-dimensional interaction and allow for new viewpoints of the virtual scenes to be generated. When using a 3-D display, the "see-through" nature of the system conveys the appropriate 3-D and visual cues to give the user a richer experience.

The proposed solution bears some similarity in concept to previous approaches. Kutulakos and Vallino[10] present an uncalibrated system that tracks fiducial points of planar surfaces in the scene and overlays the augmented video. Rekimoto[14] describes a see-through/magnifying glass augmented reality display that uses gyro sensors and detects visual barcodes in the scene. These approaches focus primarily on moving around an augmented display to interact with the real world and to superimpose virtual objects and contextual information. In contrast, the proposed solution emphasizes the interaction and design of virtual objects in the context of the real world, instead of tracking the display's relative position and orientation in the real world. Moreover, it gives the user a portable mixed-reality solution that could work with a variety of displays (e.g. notebooks, desktops, TabletPCs, PDAs, projected, heads up). This approach also gives the user better visual cues to see the traces in context (e.g. the user can see the image of his/her hand while sketching), even for uncalibrated cameras.

Figure 8 shows an example of the prototype mixed-reality system using ArcSpace. The cameras typically would be mounted directly to the back of the display; in the figure, they have been mounted on a stereo bar and moved from

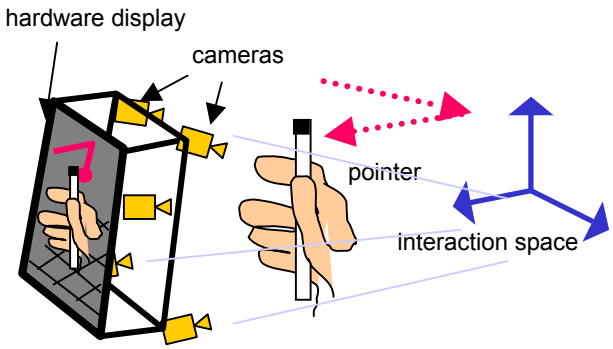

Figure 7. Improving contextual and visual cues with ArcSpace.

Figure 8. Example of prototype mixed-reality system with cameras oriented behind the display.

behind the display to better illustrate their location. The user can interact and introduce virtual annotations directly to the rabbit model located in the interaction space behind the display. The annotations (the drawn yellow hat and the green/magenta "hi" quotation) are clearly visible in the display, giving the user the feeling of a transparent display and interface. This apparent transparency helps to provide the appropriate visual context when interacting with the object. When using two cameras, the results may also be displayed as a stereo pair, as shown in Figure 9, where the virtual annotations again are automatically aligned at the correct depth relative to the object.

5. CONCLUSIONS

This paper presented a simple real-time 3-D user interface using multiple cameras geared for interaction and design. It computes multiframe correspondences in real-time, enabling this multi-dimensional quantity to serve as a form of media as well as gesture information. Through view interpolation techniques, it enables the user to change and seemingly manipulate the viewpoint of the scene even in the absence of camera calibration. The preliminary 3-D sketching application highlights its ease-of-use and applicability to a variety of 3-D applications.

Future work focuses on improving the usability of such a low-cost system for desktop augmented reality. It will be important to examine more closely the proper configuration of the cameras to convey the best 3-D experience for both capture and display. The differences among the cameras (e.g. nonlinear lens distortion, focal lengths, vignetting, color, luminance) need to be equalized to produce better synthesized views. A study of the ergonomics of such a system for extended use will obviously be necessary. Other future directions include improving the depth merging and blending of both real and virtual objects, and exploring other appropriate gestures including handwriting recognition and trainable 3-D gestures. The proposed interface serves as a good initial step towards a flexible, intuitive, and portable mixed-reality display system.

REFERENCES

1. K. Abe, H. Saito, and S. Ozawa, "3-D Drawing System via Hand Motion Recognition from Two Cameras," *IEEE Conference on Systems, Man, and Cybernetics*, October 2000.
2. A. Azarbayejani and A. Pentland, "Camera Self-Calibration from One Point Correspondence," *MIT Media Laboratory Tech Report 341*, 1995.
3. R. Bowden, T. Heap, and C. Hart, "Virtual Data Gloves: Interacting with Virtual Environments through Computer Vision," *U.K. VR-Sig Conference*, Leicester, UK, July 1996.
4. N. L. Chang, "Creating Interactive 3-D Media with Projector-Camera Systems," *SPIE Visual Communications and Image Processing (VCIP)*, San Jose, CA, vol. 5308, pp. 850—861, January 2004.
5. N. L. Chang, "Efficient Dense Correspondences using Temporally Encoded Light Patterns," *IEEE International Workshop on Projector-Camera Systems (PROCAMS)*, Nice, France, October 2003.

Figure 9. Virtual annotations in captured stereo pair for desktop mixed-reality example.

6. J. Davis and X. Chen, "LumiPoint: Multi-User Laser-Based Interaction on Large Tiled Displays," *Displays*, vol. 23, no. 5, pp. 205—211, November 2002.
7. P. Dietz and D. Leigh, "DiamondTouch: A Multi-user Touch Technology," *ACM Symposium on User Interface Software and Technology (UIST)*, Orlando, FL, pp. 219—226, November 2001.
8. I.-J. Lin, "Active Shadows: Real-Time Video Segmentation within a Camera-Display Space," *International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*, Lisbon, Portugal, April 2004.
9. G. Klinker et al., "Confluence of Computer Vision and Interactive Graphics for Augmented Reality," *Presence: Teleoperators and Virtual Environments*, vol. 6, no. 4, pp. 433-461, 1997.
10. K. N. Kutulakos and J. R. Vallino, "Calibration-free Augmented Reality," *IEEE Trans. Vis. Comp. Graph.*, vol. 4, no. 1, pp. 1—20, 1998.
11. A. Pentland, "Smart Rooms," *Scientific American*, vol. 274, no. 4, pp. 68—76, April 1996.
12. T. Poston and L. Serra, "Dextrous Virtual Work," *Communications of the ACM*, vol. 39, no. 5, pp. 37—45, 1996.
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," *SIGGRAPH*, Orlando, FL, July 1998.
14. J. Rekimoto, "NaviCam: A Magnifying Glass Approach to Augmented Reality Systems," *Presence: Teleoperators and Virtual Environments*, vol. 6, no. 4, pp. 399—412, 1997.
15. C. M. Schmandt, "Spatial Input/Display Correspondence in a Stereoscopic Computer Graphics Workstation," *Computer Graphics*, vol. 17, no. 3, pp. 253—262, 1983.
16. B. Ullmer and H. Ishii, "The MetaDESK: Models and Prototypes for Tangible User Interfaces," *ACM Symposium on User Interface Software and Technology (UIST)*, Banff, Alberta, Canada, pp.223—232, October 1997.
17. A. Wilson, "TouchLight: An Imaging Touch Screen and Display for Gesture-Based Interaction," *International Conference on Multimodal Interfaces (ICMI)*, State College, PA, pp. 69—76, October 2004.