Light Widgets: Interacting in Every-day Spaces

Jerry Alan Fails, Dan Olsen, Jr.
Computer Science Department
Brigham Young University
Provo, Utah 84602
{failsj, olsen}@cs.byu.edu

Abstract
This paper describes a system for ubiquitous interaction that does not require users to carry any physical devices. In this system, the environment is instrumented with camera/processor combinations that watch users while protecting their privacy. Any visible surface can be turned into an interactive widget triggered by skin-colored objects. Light widgets are tied to the XWeb cross-modal interaction platform to empower them with interactive feedback.

Keywords
Ubiquitous computing, computer vision, cross-modal interaction

INTRODUCTION
Following Moore's Law, computing continues to evolve rampantly. This has caused the ratio of computing devices to humans to drastically increase. In this socio-technical setting, the desktop has become too restrictive for most situations where people work and play. This has led to extensive research in the realm of Ubiquitous Computing [12, 13, 11, 1, 10, 16]. Unfortunately, much of the research involving ubiquitous computing requires the user to wear or carry with them some sort of physical device. Such devices provide user identity, detect tags in the environment, detect user gestures, or provide display capabilities. However, carrying a physical device is inconvenient. The problem is to create a low-cost, versatile, adaptable and integrated ubiquitous system that can be used in any indoor space without carrying anything. To accomplish this, we mount a series of cameras that can watch what the user is doing and perform interactive behaviors based on the surfaces the user touches.

Ubiquitous Computing and Augmented Reality
Traditionally, ubiquitous computing and augmented reality have had many common goals. Projects like *NaviCam* [11] and *Cyberguide* [1] attempt to view the real world while touring through it and augmenting the view with digitized information. There are yet other systems like the Gesture Pendant [13] that require the user to carry a device with them that does gesture recognition, but still manipulate digital data. We are not trying to augment the world with information, but integrate interactivity into the physical world. We strive to instrument the environment with inexpensive devices that allow users to manipulate digital information.

Instrumenting the environment for ubiquitous interaction is not a novel idea. Many systems have used environment tags, both electro-magnetic and visual, to be able to locate users or objects within the environment, and set values according to their placement [7, 14]. The commonly used electro-magnetic tracking system used is RFIDs (Radio Frequency Identification) [8, 14]. RFID tracking requires the user to carry an RFID with antennas scattered throughout the environment, or to carry an antenna and scatter RFID tags throughout the environment. Neither of these options satisfies our goal of not requiring the user to carry any physical device. Instead, we took the visual approach by using simple computer vision. We chose this approach because it does not require the user to carry anything and it is easy to dynamically reconfigure.

In this project, we have geared our efforts towards simple ubiquitous computing. In so doing, we have not ignored the vital issue of user feedback, which is crucial to all computing systems. Feedback in our system is achieved by integration with the XWeb system. This affords instant cross-modal interaction [9]. By tying our system to XWeb, all XWeb interactive clients and servers are available to provide feedback. Currently there are XWeb clients implemented for: speech, wall, projector, TV and desktop interactivity; XWeb servers include X10 and desktop environments.

Light Widget System Overview
Light widgets are predefined widgets that allow users to select values with their hands, as in Figure 1. Triggering of a light widget occurs when skin is detected on the light widget. This differs from gesture-based systems like Gesture Pendant [13] because value selection is based simply by skin color detection in the light widget regions.

IUI'02, January 13-16, 2002, San Francisco, California, USA.

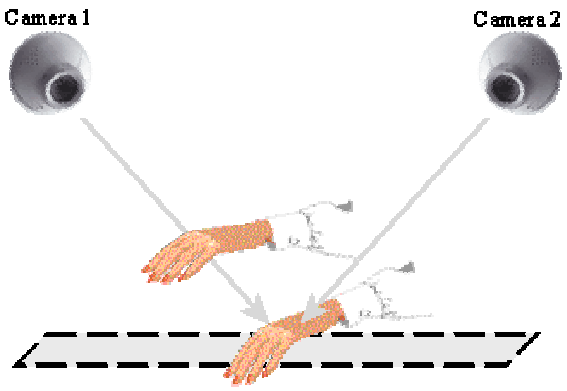

Figure 1 — Multi-camera detection

The simple skin detection technique we use demands a multi-camera system for correct detection of when a user is using a light widget. If, for example, we only used Camera 1 in Figure 1, then both hand positions would activate the light widget represented by the dotted area. By triangulating with both cameras, only the lower right-hand triggers detection. This multi-camera system decreases the amount of false-positives.

Using light widgets, a user could, for example, control the volume of his/her stereo using a slider-type light widget placed along the side of a desk. The user could then slide their hand along the side of the desk until the desired volume was reached. Just as easily, a mechanic could control the height of the car he is working on by placing a light widget along the wall, on the floor or across the top of a stationary toolbox. A light widget could be advantageous in this setting as the mechanic may not want to remove himself from the place he is working just to adjust the height of the vehicle. Another user might create two light widgets on the headboard of her bed and one on the nightstand next to it. She could create a light widget to turn the TV off and on by touching the corner of the nightstand, set the volume by touching the pole of the headboard, and change the station by moving her hand across the top of the headboard. This example is illustrated in Figure 2.

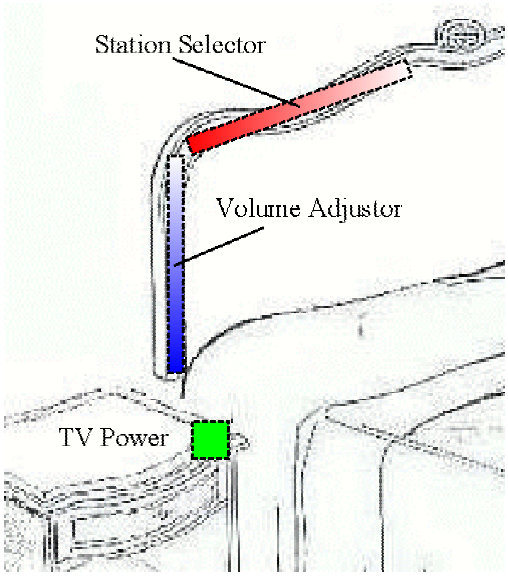

Figure 2 — Bedroom/headboard example; TV controlling light widgets

The above examples show the contrast between our approach and tagging. Picking up an object such as a tag or antenna to change the volume or raise a car is much less convenient than simply touching a spot with your hand. In addition, any object small enough to carry conveniently can get lost in a shop or on a desk. Another advantage of using computer vision and cameras is we enable multiple light widgets to be monitored by one camera pair. This supports diverse interactions at low cost.

The above examples can also be compared with a system like Gesture Pendant. First, to use the Gesture Pendant system, the user would have to have the physical Gesture Pendant device with them. Then the interaction would include using speech to choose an interactor, followed by a gesture to modify the selected interactor. Using light widgets the user can set up interactors anywhere and select and adjust that interactor simultaneously, by simply moving his/her hand to adjust the value. The interaction using light widgets is more simplistic because the user need not have a physical device with her/him, nor need s/he previously specify which interactor the gesture will modify.

Light widgets are designed to easily fit into the user's environment and provide access to technology at low cost. In the same way as motion sensor lights are used for patios and yards, light widget systems blend easily into the environment and facilitate access to electronic resources. Each system (pair of cameras) can manage several light widgets, as well as manipulate varied value types of different objects. Although light widget sensor's versatility in scale and range greatly exceed motion sensor lights, the comparison displays how the placement of cameras, similar to motion sensor lights is trivial, unobtrusive and practical.

There are several issues that need to be addressed in creating such a system. These issues are:

- What kinds of interactions are possible using cameras?
- Can the technology be cheap enough to be ubiquitous?
- How do we provide feedback since cameras are input-only devices?
- How do we ensure privacy with our use of cameras?
- How does a user configure light widgets and integrate them with other interactive facilities?

Possible interactions using cameras

The light widget system currently implements simple controls for setting atomic data types: switches, numbers, dates and times. Users can perform all interaction by placing a hand on a surface area configured as a light widget. Button light widgets allow things to be turned on and off, while linear and circular light widget areas control

continuous values. The camera processing for these tasks is simple skin blob detection.

Inexpensive technology

We propose a modification to Figure 1 by adding micro-controllers to each individual camera. This modification allows each camera to process the image locally and then report detected light widget values to a server. Each micro-controller will be responsible for two things: skin-blob detection and light widget value approximation. To minimize costs these micro-controllers cannot be very powerful. We need to minimize the skin-detection processing as well as the light widget evaluation. The functionality of the micro-controller is shown in Figure 3. When manufactured in quantity, each computer/camera combination should cost about the same a motion sensor light.

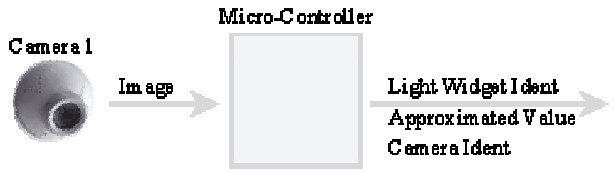

Figure 3 — Micro-controller functionality

In our demonstration prototype we did not use micro-controllers on each camera. Instead we used multiple threads on a single PC, one for each camera and one for the server. As will be shown later, the algorithms used for these two computations are simple enough to download onto small, inexpensive micro-controllers.

Providing feedback

Just like traditional GUIs, there must be some form of feedback in response to user gestures. Some ubiquitous interaction projects have used projectors as their means of feedback [15, 17]. Projectors can be very expensive which defies our goal of low-cost ubiquitous computing. Even though cameras are input-only devices, light widgets must provide some feedback mechanism to the user. For example, light widgets must provide feedback when a user changes the optimal temperature on a thermostat linear light widget. With light widgets, the user does not change a physical object and the new setting is not always physically manifest. Hence the user must rely on some other mode of feedback. We provide this feedback by integrating light widgets with the XWeb cross-modal interaction platform [9].

XWeb provides subscription services to data, which enables interactive clients to monitor data changes. Through this subscription mechanism, any number of XWeb interactive clients can be slaved together. Any interaction in one client results in changes being propagated to all other clients viewing the same data. We have implemented XWeb clients for projectors, TVs, the traditional desktop, wall pens, laser pointers and speech. This means that by integrating with XWeb we can allow instant access to any of these means of feedback. If there is a TV in the room where the light widgets are being monitored, when a light widget is used, the manipulated data can report as changed on the TV screen. If we are in a more obscure location, the speech client offers a fitting feedback mechanism for light widgets. This capturing of devices already in the user's environment continues to meet our goals of ubiquitous computing. By using the cross-modal features of XWeb, we allow projectors to be a manner of feedback, while still providing several other less expensive feedback mechanisms.

Ensuring privacy

Our bed/headboard TV controller example illustrates the need to address the issue of privacy. If our light widget cameras sent images out of the room, privacy would be violated and people would feel very uncomfortable. Hudson's response to this privacy issue is to obscure people so they cannot be personally identified [6]. This is not suitable for our interactive needs. Our solution is to have each camera process the image locally and report its conclusions to a server. Each camera need only transmit the camera identifier, the light widget identifier and its approximated selected value, as shown in Figure 3. If the images never leave the camera, then the privacy problem is vastly reduced. It is still possible to detect interactive activity, but nothing else. If the user does not activate a light widget then no information leaves the camera. This solution allows light widget cameras to be used in personal spaces, like the bedroom, where image transfer is inappropriate. We think of them not as cameras but as "optical interactive gesture detectors". To users in the bedroom this difference is important.

Configuring light widgets

A user needs to be able to easily configure a light widget. We created a simple application that takes a snapshot from each camera and then allows the user to draw the light widgets onto the snapshots. The user can then create a link between an existing XWeb interface and the light widget.

A problem with this configuration approach is getting the snapshot images from the cameras, without violating privacy. One answer is to have a USB or other inexpensive connection attached to each camera to retrieve images. Although other configuration methods could be used, by using a USB connection, the user knows whether images are leaving the room or not, as a physical device must be plugged in for external image transfer to occur.

LIGHT WIDGET IMPLEMENTATION

Having defined our goals for light widgets, we must address the implementation issues. The two key issues are image processing (detecting basic interaction) and XWeb integration.

Detecting Basic Interaction

Since light widgets are selected using hands, an efficient skin-detection algorithm is required in their implementation. Much research continues to be done in the area of skin-detection [19, 5, 4, 2]. These techniques vary in accuracy and processing requirements. By weighing the computing cost relative to the efficiency we decided to use a mixture of the Bayesian and Parzen window algorithms, based upon Zarit, Super and Queck [20]. This algorithm requires a set of training examples to be fed to a color training application. We use hue and saturation for skin-detection, because it is commonly accepted that hue and saturation are more robust to illumination differences and different skin colors. We quantize hue and saturation values into a 60x60 look-up table. Using Bayesian probabilities we compute a skin/no_skin value for each cell in the table. Using this algorithm, skin detection becomes a simple matter of indexing into this table with hue and saturation values. It is now simple to classify each pixel as skin or not skin. This trivial algorithm affords great speed, low memory and approximately 85% accuracy. This skin color detection algorithm allows less expensive hardware to be used and achieves comparable results to the more complex skin-detection algorithms. We get further speedups by not considering all pixels in the camera image, but only those in the area of each light widget. Thus we look at less than 10% of the pixels in an image.

All light widget interaction is based upon skin-detection. As shown in Figure 1, light widgets can be set up in any region seen by two cameras. Light widgets are triggered by skin colored objects that are placed on the surface of a light widget's visible area. Each camera finds skin color blobs, computes their center of mass and then evaluates an approximated light widget value based on that center of mass. Each camera reports their approximated values to a server that resolves the votes for each approximated value.

The server will set the value if and only if two or more cameras report a similar value. So, in the case of Figure 1, the hand directly on the light widget will be detected as a selection as both cameras will report similar selected values. The left hand in Figure 1 will not trigger a selection because Camera 1 would report a different selected value than Camera 2. This voting algorithm is computationally trivial and meets our requirement of "no images leave cameras". False-positives are greatly decreased by using multiple camera perspectives. Conversely, the approximate 85% accuracy is effectively increased because a false-positive can only actually occur if similar errant light widget selected values are reported by at least two different perspectives. This simple, multiple camera system is configurable for diverse environments and inhibits the problem of false-positives.

Integration with XWeb

As stated before XWeb was chosen because of its cross-modal capabilities. This cross-modal interaction is made possible by the ability to subscribe to common data. For example, instead of just projecting the information back onto a desktop, as in [17, 14, 15], we can synchronize an XWeb speech client [9] to a light widget system and the light widgets can audibly report their changed values. Alternatively, by using XWeb, we could have the information display in an XWeb view on an available TV. This interactive feedback mechanism notifies the users when their light widgets have been activated. By using XWeb's cross-modal functionality, we amplify the feedback space available to us.

Since we are using XWeb as our interface between setting values and light widgets, we need to understand more about what interactors or predefined widgets, XWeb has, and also, understand what setup need to be done so that light widgets can interface with XWeb values.

XWeb has several types of interactors. The atomic interactors that both XWeb and light widgets can manipulate are: Enumerations, Numbers, Dates and Times. XWeb also has a Text interactor. However, it is interactively not feasible to write in a text box using our light widget techniques. Numbers, Dates and Times are interactors that have inherent ordering, and so we allow slider-type light widgets (linear and circular light widgets) to control these types of values. Enumerations, in general, do not have an explicit ordering, so Enumerations can only be manipulated by button-type light widgets.

An XWeb interactor has an XLoc (similar to a URL) that references the data that the interactor is manipulating. To integrate light widgets with XWeb, we need to extract the interactor type and its XLoc from an existing XWeb interface.

THE LIGHT WIDGETS

Three light widget type have been implemented: button, linear and circular light widgets. Button and linear light widgets are similar to their GUI (Graphical User Interface) counterparts: buttons and sliders. The circular light widget is a circular slider. Figure 4 shows the light widget setup application with an example of each widget type.

Button Light Widgets

Button light widgets have the same "feel" as GUI buttons. There are single-value buttons and toggle buttons that have different *on* and *off* values. A single-value button simple associates a data reference and a value with the light widget's visual region. When the user touches that region the data reference is set to the value. This is a simple switch mechanism. Using three of these light widgets, a radio group of three items can be constructed in three adjacent places or on three related objects.

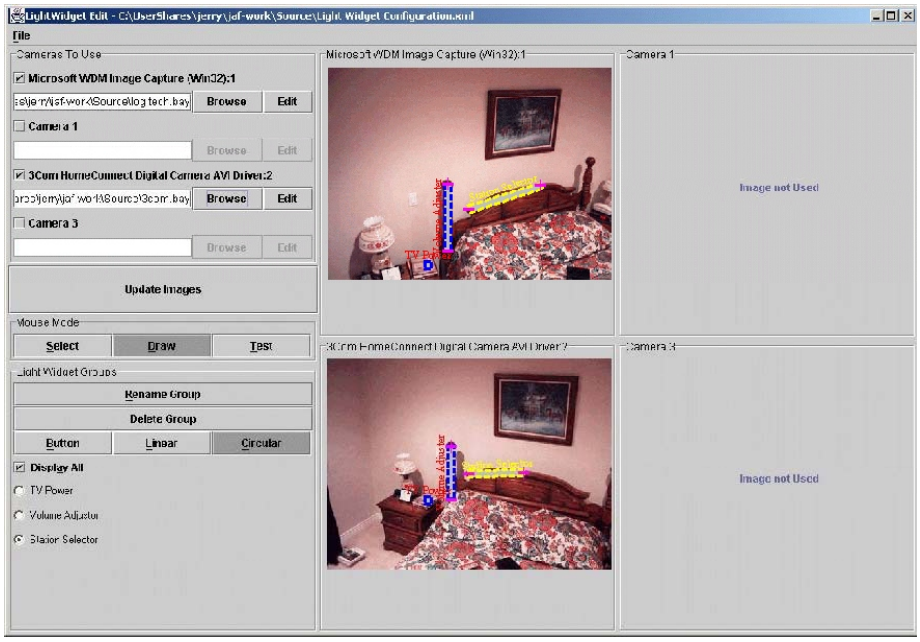

Figure 4 — Light Widget Configuration Application

A toggle button has a data reference, two values and a visual region for each camera. Placing one's hand on the visual region will toggle the data between two values.

To set up a button light widget a user need only use the light widget setup application and draw rectangles on the two camera images where the button light widget should appear in each image. The user then selects the light widget and displays its properties. The property edit box is shown in Figure 5. If this were a newly created button light widget, the URL and XLoc would be empty and the user would need to provide the link between the light widget and the desired XWeb interactor.

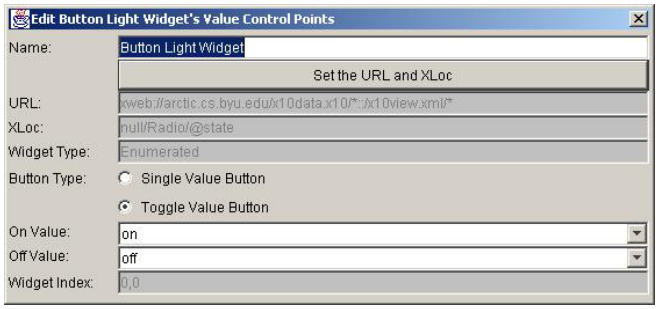

Figure 5 – Button light widget property edit box

The link between the XWeb atomic value and the light widget is the most critical property to setup. To provide this link to an XWeb interactor, the user presses the "Set the URL and XLoc" button. The user then sees the message in Figure 6, which prompts them to go to the XWeb GUI interface, select an XWeb interactor and then hit OK. The light widget configuration system then captures the necessary XWeb information and stores it with the light widget.

Figure 6 – Instructions for creating an XWeb link

For example, the user could go to a home automation page in the XWeb GUI and select the radio, as shown in Figure 7.

Figure 7 – XWeb home automation interface

After the user selects the OK button, the button properties of URL, XLoc, widget index along with default values for *on* and *off* values are set. This link setup is the same for all light widgets.

These virtual buttons are of great use in an environment. Power on/off pairs for any electrical device can be created virtually without rewiring the light switches of a house. Obviously, since this system uses normal cameras, button light widgets do not work for room light switches, as it is impossible for the cameras to detect skin if the room is completely dark, however, their usefulness for lamps, TVs, stereos and other electrical devices is unlimited. The ambient light problem might be overcome with infrared cameras, much like in the Gesture Pendant [13], which can readily detect skin. However, we used only normal, visual light cameras.

Linear Light Widgets

Linear light widgets interact with a range of values. A linear light widget must have maximum and minimum values. To manipulate a thermostat, a linear light widget could be set up to have a minimum value of 60 and maximum value of 80 and could be placed along the casing of a doorway. A user could slide his/her hand along the doorway casing until the desired temperature is set.

The properties edit box is shown below for an existing linear light widget that manipulates an XWeb thermostat. The same XWeb link setup as explained for button light widgets is used to setup the URL and XLoc for the wakeup temperature on this XWeb thermostat controller. The property edit box for such a thermostat widget is shown in Figure 8.

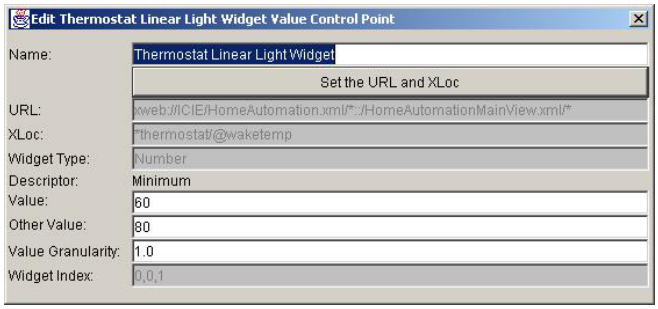

Figure 8 – Linear light widget property edit box

Linear light widgets also have granularity. Since linear light widgets have a range, the value between approximated selection values is a real value between the maximum and minimum values. This exactness is often unnecessary and sometimes completely undesired. By adding a granularity value, the user can decide how fine or coarse the approximation should be. In the thermostat example above, the granularity is set to one, which signifies that the approximation will be evaluated to every 1 degree. If the granularity were 2 and the min and max values were still 60 and 80 respectively, the widget would evaluate to one of the even numbers between 60 and 80 inclusive. Granularity accepts a real number, so if the granularity were set to 0.5 then the cameras would approximate selection values to the nearest half-degree. Granularity is also important in multi-camera voting. With the infinite granularity two cameras would rarely report the same value for a given light widget. Using a more coarse granularity resolves this excessive sensitivity.

Circular Light Widgets

Circular light widgets provide round control surfaces similar to knobs or clock surfaces. One of the problems with a circular space is defining the angular origin. Circular light widgets add two properties beyond the linear light widgets — an initial angle and a direction. For example, Figure 9, shows a circular light widget from two angles, the initial and on each is unique, but identifies the starting and stopping point of value allocation around the circle. Each light widget must also have a direction: clockwise or counter-clockwise. This model assumes cameras will not have mirrored views, direction is assumed to be uniform for both perspectives.

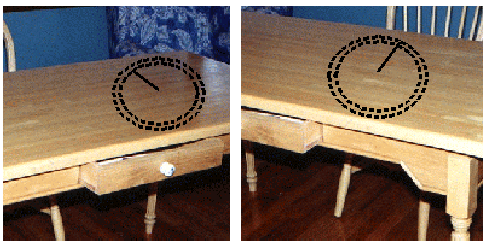

Figure 9 – Circular light widgets

A practical use for a circular light widget could be to create a sprinkler start time controller. In Utah, we have been asked to only water our lawns from 9pm to 9am. By setting the start angle to the 9 o'clock position on a clock, the circular light widget is easy and intuitive. A granularity of 15 could be imposed to only allow start times every quarter of an hour. The property edit box for such a circular light widget is shown in Figure 10.

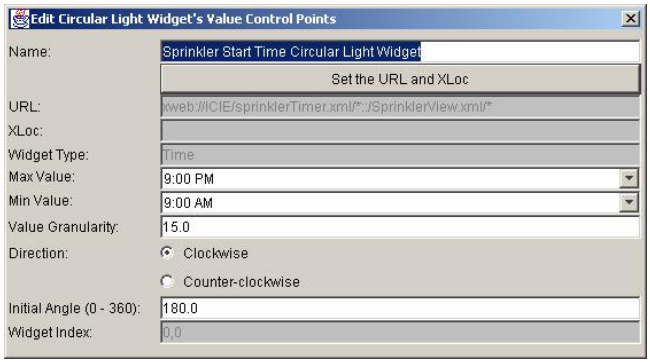

Figure 10 Circular light widget property edit box

THE LIGHT WIDGET PROTOTYPE

Our light widget prototype uses USB cameras connected to a personal computer. The system is relatively low-cost. Two USB cameras were used in our implementation and can be purchased for under $100. Connectivity costs are low, as each camera reports only its ID, the light widget's ID and the estimated value for the widget. Each camera will need a micro-controller to perform the image processing and value approximation, but this is also low-cost as the processing power required is minimal as the image processing algorithms used are trivial.

CONCLUSION

We have met our goals for ubiquitous interaction using multiple inexpensive cameras to sense user hand movements. Using these cameras we can create new light widgets simply by drawing them on snapshot images from each camera. We ensure privacy by having each camera emit only its votes for light widget values. We resolve multi-camera integration by simple value voting rather than 3D geometry. Skin detection is performed by a simple

lookup of quantized hue and saturation values. We believe such a system can provide interaction anywhere.

REFERENCES

[1] Abowd, G.D., Atkeson, C.G., Hong, J., Long, S., Kooper, R., and Pinkerton M. "Cyberguide: A mobile context-aware tour guide." ACM Wireless Networks, 3:421-433, 1997.

[2] Crowley, J.L., Bérard, F., and Coutaz, J. "Finger Tracking as an Input Device for Augmented Reality." IWAGFR '95: Zurich, Germany (June 1995).

[3] Fitmaurice, G., Ishii, H., and Buxton W. "Bricks: Laying the Foundations for Graspable User Interfaces." *Proceedings of CHI '95* (Denver CO, May 1995), ACM Press, 442-449.

[4] Ghidary, S.S., Nakata, Y., Takamori, T. and Hattori, M. "Head and Face Detection at Indoor Environment by Home Robot." *Proceedings of ICEE2000* (Iran, May 2000).

[5] Heap, A.J. "Real-Time Hand Tracking and Gesture Recognition Using Smart Snakes." Interface to Human and Virtual Worlds: Montpellier, France (June 1995).

[6] Hudson, S.E and Smith, I. "Techniques for Addressing Fundamental Privacy and Disruption Tradeoffs in Awareness Support Systems." Proceedings of the ACM on Computer Supported Cooperative Work (Boston MA, November 1996).

[7] Ishii, H. and Ullmer, B. "Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms." *Proceedings of CHI '97* (Los Angeles CA, April 1997), ACM Press, 234-241.

[8] Koller, D., Klinker, G., Rose, E., Breen, D. Whitaker, R and Tuceryan M. "Real-time Vision-based Camera Tracking for Augmented Reality Applications." *Proceedings of the Symposium on Virtual Reality Software and Technology* (VRST-97), Lusanne, Switzerland, Sept 1997, 87-94.

[9] Olsen, D.R., Jeffries, S., Nielsen, T., Moyes, W. and Frederickson, P. "Cross modal Interaction using Xweb." *Proceedings of UIST '00* (San Diego CA, November 2000).

[10] Olsen, D.R. "Interacting in Chaos." *Interactions*, Sept 1999.

[11] Rekimoto, J. and Katashi Nagao. "The World through the Computer: Computer Augmented Interaction with Real World Environments." *Proceedings of UIST '95* (Pittsburgh PA, November 1995), 29-38.

[12] Schilit, B.N., Adams, N. and Want, R. "Context-Aware Computing Applications." In Proceedings Workshop on Mobile Computing Systems and Applications. IEEE, December 1994.

[13] Starner, T., Auxier, J. and Ashbrook D. "The Gesture Pendant: A Self-illuminating, Wearable, Infrared Computer Vision System for Home Automation Control and Medical Monitoring." *International Symposium on Wearable Computing* (Atlanta GA, October 2000).

[14] Underkoffler, J., Ullmer, B. and Ishii, H. "Emancipated Pixels: Real-World Graphics in the Luminous Room." *Proceeding of SIGGRAPH '99* (Los Angeles CA, 1999), ACM Press, 385-392.

[15] Underkoffler, J. and Ishii H. "Illuminating Light: An Optical Design Tool with a Luminous-Tangible Interface." *Proceedings of CHI '98* (Los Angeles CA, April 1998).

[16] Want, R., Schilit, B., Adams, N., Gold, R., Petersen, K., Goldberg, D., Ellis, J., and Weiser, M. "The ParcTab Ubiquitous Computing Experiment." Xerox Parc technical report. http://citeseer.nj.nec.com/535.html

[17] Wellner, P. "Interacting with paper on the DigitalDesk." *Communications of the ACM*, 36(7):86-96, July 1993.

[18] Wisneski, C., Ishii, H., Bahley, A., Gorbet, M., Braver, S., Ullmer, B. and Yarin P. "Ambient Displays: Turning Architectural Space into an Interface between People and Digital Information." Springer Verlag, Feb 25-26 (1998).

[19] Yang, M.H. and Ahuja, N. "Gaussian Mixture Model for Human Skin Color and Its Application in Image and Video Databases." *Proceedings of SPIE '99* (San Jose CA, Jan 1999), 458-466.

[20] Zarit, B.D., Super, B.J. and Quek, F.K.H. "Comparison of Five Color Models in Skin Pixel Classification." ICCV '99 International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems (RATFG-RTS '99), Corfu, Greece, Sept 26-27 (1999), 58-63.