24

Physical Interaction

Most of this book has focused on interaction on a screen. However, much of the living, working and playing in this world does not occur at a desk or in front of screen. Computing also needs to be available off the desktop where people live their lives. This is part of Mark Weiser's vision of *ubiquitous computing*.[1] In moving out into the physical world we need to revisit the model-view-controller architecture, as shown in figure 24.1. In the physical world where people are not necessarily working, the information about the user interface is broadcast into the space rather than presented on a focused screen. We must take into account how the information will make its way into the physical world and how the user might understand it.

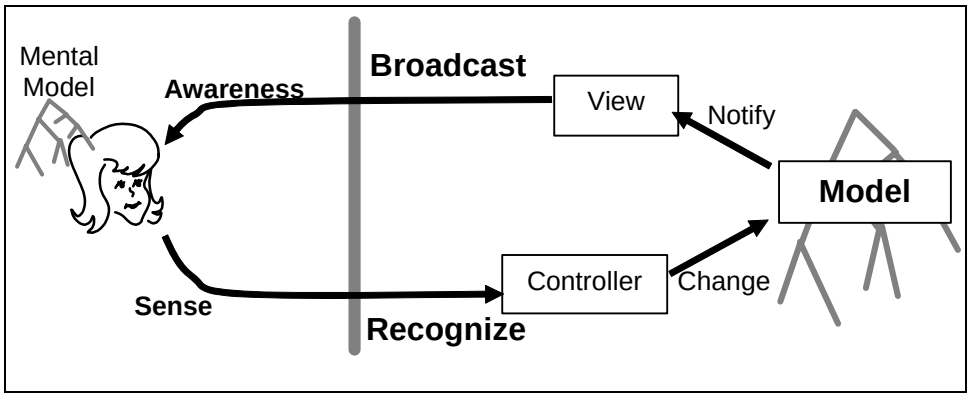


Figure 24.1 – Physical world model-view-controller

A particular problem in the physical world is awareness. If the user is not aware of the information being presented, then there is no interaction. However, if the user does not want to interact right now, then a system that demands awareness becomes annoying and will be turned off. Broadcasting of information also involves privacy and politeness issues. People are frequently not alone. Information that others should not see, should not be broadcast. Information that others do not want to know can be annoying to them. People who talk loudly to

the cord microphones of their cell phones in public are generally considered rude by their neighbors.

Presentation of information into the physical world must first have a channel for communication. The limitations of humans dictate that this channel must be visual, audio or, to a limited extent, touch. Broadcast of visual information can violate privacy, but if the images are not moving, a lot the information can be ignored by those who are disinterested. Visual information requires the user's attention in order to communicate. The management of awareness and interuptability is key to visual broadcasting. Interactivity requires that user input and displayed information must be calibrated to each other. This is more difficult in the physical world than on a screen. Audio allows a user to receive information without paying direct attention. However, audio is very difficult to ignore and has low bandwidth. Broadcasting of audio into a shared space can be very distracting. Tactile information is limited to awareness tasks such as the vibrating of a cell phone.

The largest challenge to interacting the physical world is sensing what the user is doing and what the user wants done. The user's interactive expressions of intent not be encumbered by too many devices and be as natural as possible within the user's environment. Avoiding encumbrance poses a sensing problem. We need devices and sensing techniques that fit into the user's environment without making them feel like they have been "instrumented". Naturalness poses the problem of ambiguity. If the gesture is natural to the user, it may be accidentally expressed.

Fitting interactive computing into the lives of people poses challenging usability problems. For this chapter, however, we will focus on the technologies that are available. First the techniques for presenting information and managing user's awareness will be discussed. This will include techniques for detecting when a user can be politely interrupted by new information. Secondly there will be an extensive discussion of the technologies and software for sensing what the user is doing and the user's intent. These techniques will rely heavily on the machine learning algorithms in Appendix A3.

Controlling light and awareness

Hiroshi Ishii introduced the concept of the I/O bulb[2]. The idea is to replace light bulbs with a digital projector/camera combination. We will discuss camera interaction later in this chapter. The concept of replacing every light bulb with a projector is an interesting one. Not only can the projector provide normal

illumination but it can control that illumination in interesting ways to enhance the living/working experience. For our purposes the projector allows us to present information to the user. Projecting information into a room involves five basic problems. The first is to get enough brightness and contrast so that the information is visible. The problem is that room illumination varies widely and the illumination suitable for comfortable living tends to wash out many projectors. To overcome this, projectors are sold with very high brightness.

Projector brightness is the second problem. Very bright lights generate lots of heat which must be dissipated. Heat leads to noisy fans. Mitsubishi has introduced a fanless projector using LEDs for the light, but the brightness is not very high. In addition, accidentally looking into a very bright light can be painful. This "sun spot" problem from inadvertently looking into a projector can be addressed by shining down from the ceiling. People rarely look straight up.

The third problem is projector/surface placement. Traditionally a projector is placed perpendicular to a very flat, specially prepared screen. This works fine for conference rooms but not so fine for use in the general environment. People want their walls where they want them and want the projectors to be unobtrusive. The Office of the Future[3] project created techniques for sampling the environment and then warping the projected images so that they appeared correct on any light reflective surface. The problem with their system was that the current position of the user's eyes must be known to correctly compensate for the irregularities of the surface. Instrumenting the user's head seems a little intrusive. GVU-PROCAMS[4] assumes that the projection surface is flat and then provides algorithms to compensate for the distortion cause by a projector that is not perpendicular to the surface.

The fourth problem is user shadowing. If the user is going to live or work in the same environment as the projected information then the user will occlude the projector. This is increasingly likely as the user gets closer to the projection surface. One solution is to project from behind the surface, as with many large-screen televisions. This has the two-fold problem of requiring additional space for the projector/mirror and a special translucent surface for projection. Rear projection surfaces are frequently not robust working surfaces. GVU-PROCAMS uses multiple projectors from oblique angles to eliminate shadowing (figure 24.2). This is the same as using multiple lights in a workspace eliminate shadows. Perhaps if Ishii's vision of the cheap I/O bulb becomes reality a ceiling might have many small projectors to resolve shadowing. The use of many overlapping projectors may also solve the brightness problem.

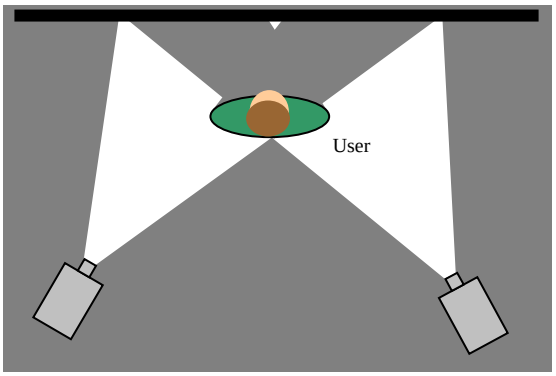


Figure 24.2 – Multiple projectors to eliminate shadowing

The fifth problem is the necessary display size and resolution. The eye does not really have a pixels-per-inch resolution. The resolution of the eye is in *degrees of visual arc*. A readable piece of text needs to be about 0.7 degrees high and at least 10 pixels. The degrees of height relate to the resolution of the retina and the pixel height relates to clear presentation of the text. The actual pixels-per-inch resolution required depends upon the viewing distance. A good rule-of-thumb for text height is 0.008 times the distance to the screen. When presenting in a physical environment where users are moving around the viewing distance varies and thus the size of displayed objects must vary to maintain the same height in visual arc. There are really only two choices in high movement situations. The first is to pick an average viewing distance and the second is to track the position of the user.

Polite Presentation

One of the complaints about computing technology is that it is so intrusive. When moving into the physical world we want computers that behave more politely. Researchers have addressed these issues in two ways: 1) ambient displays and 2) user interruptability.

An ambient display is one that is readily ignored but can be perceived peripherally. This is similar to a mother monitoring the children in the other room by the noise they make. She is not really listening to what they say, but becomes aware when crying starts or a period of dead silence occurs. Ambient displays can take a variety of physical forms and are not necessarily images. The Live Wire[5] is just a plastic cord attached to an electric motor. The motor speed is controlled by the amount of traffic on the network. As the traffic grows the motor

goes faster cause the cord to whip around. The cord allows operators to be aware of the traffic without paying direct attention to it. If the network goes down, operators are immediately aware of the fact even though they were not paying attention. A similar technique is to control the rate of water dripping into a pool. The reflection off the water is projected onto the ceiling where the value being presented is seen peripherally as expanding ripples of water. The Bubble Wall[6] presents information by inserting bubbles into columns of water. Because bubbles rise slowly, it is possible to time the bubble generation so that images can be displayed in the rising bubbles. Ambient displays have become the subject of many art installations where information is presented in ways that integrate naturally into the environment.

An interesting variant on the ambient display are the "glance displays" designed by Vertegaal[7]. The idea is that the display only becomes active or generates foreground information when the user is actually looking at it. "Glance" detection exploits the fact that the retina is highly retroreflective in infrared. An infrared camera is surrounded by infrared LEDs. When the user looks at the object on which the camera is mounted, the light from the LEDs is reflected straight back to the camera. In the camera image the eyes appear as two glowing spots, but only when the eyes are actually looking at the region of the camera.

Another way to adapt the display of information to the user's life is to understand the interruptability of the user. Polite people do this automatically by checking to see if someone is busy and only interrupting when something is very important or the other individual appears available. A polite computer would also only interrupt when appropriate. The problem is to provide the computer with a way to determine when it is appropriate to interrupt the user. Fogarty and Hudson, et. al.[8] performed an experiment to see when it is appropriate to interrupt. They placed cameras in the offices of 4 subjects and captured everything that was going on for a total of 600+ hours. During this period the capture hardware would randomly ask the user how interruptible they were on a scale of 1-5. This happened approximately twice per hour. In addition, 40 other people viewed the video and made independent judgments about interruptibility. This annotated the video with evaluations of when the subject could be reasonably interrupted.

Coding staff were then assigned to view video segments and evaluate when various activities occurred including: is there a guest in the room?, are they talking?, is the occupant writing or working at the computer?, is the door open or closed?, is the occupant reading or on the phone? These annotations were

intended to simulate a possible sensor that a computer could use to detect interruptibility. The authors then built a machine learning model that could predict interruptibility from the simulated sensors. The surprising result was that interruptibility could be detect 76% of the time just be sensing whether or not anyone was talking. A presentation system that only interrupts when nobody is speaking would be much more polite than current systems.

Watching the User and the World

Given an appropriate presentation system the more difficult problem is the sense what is going on the world and what the user wants the computer to do. We have already seen glance and speech sensors. There are many others. One of the most powerful sensors for observing the world is the camera. With the advent of CMOS cameras this sensor has become relatively cheap. Sensing the world through cameras is generally less obtrusive than other sensors because users generally do not wear anything special and natural objects can be used as input devices. Camera sensor software can be reconfigured so that they can detect many kinds of things. However, cameras do have some disadvantages. One of the most important is privacy. People are uncomfortable having many parts of their lives watched by a digital camera. A second issue is occlusion where people and things get between the camera and the sensing target. Lastly, cameras do not work in the dark.

Image processing fundamentals

Before reviewing many of the types of interactive techniques that have been built around cameras, it is useful to present some fundamentals of image processing. For interactive work we can ignore many of the concepts of image processing, which will simplify our discussion. On the other hand interactive techniques impose rather stringent restrictions on time. For a technique to feel interactive we must be able to process an image and draw a conclusion within 200 milliseconds. That is very quick for many image processing techniques. Throughout this discussion we will rely heavily on the point algorithms in Appendix A1.3 and the machine learning algorithms in Appendix A3.

Interactive detection requirements

Interactive use of cameras generally boils down to three questions.

1) Is target T found in region R of the camera image? The region of the image may be the whole image or just a small portion. Are the user's hands on the keyboard? Is there a user in the room? Is the telephone on or off of the hook?

Did the user just touch the printer? Has the "output elf" been placed on the printer, on the screen or on the projector? These all involve looking at the pixels in some region of the image (keyboard, phone, printer, screen, projector or room) and deciding if the target (user, hands, finger, phone handset or "output elf") is present. Targets can be people, body parts or physical objects being moved by the user.

2) At what locations (X,Y) is target T found in the image? These are tracking questions. Where are the user's hands? Where on the desk are the user's fingers? Where is the user standing in the room? Where is the cell phone that the user is holding? The problem is simpler with only one target. However, there are many cases with multiple targets (two people, ten fingers, multiple telephones).

3) Is movement occurring and if so in what direction? People frequently interact by movement. We want something to happen when things change and in the physical world change is movement. We may want to know if the user is waving and if so, are they waving up and down or side to side?

In the remainder of this section we will discuss a variety of useful features that can be drawn from images and then used to find or track objects in the physical world. Many of the terms here are the same as found in the image processing literature but they refer here to fast and somewhat "dirty" approximations. In the interactive world speed is everything and we frequently sacrifice mathematical accuracy to get it.

pixel features

Because most of our techniques will involve machine learning, we will need techniques for extracting features from images. With a vector of features we can then apply a machine learning algorithms to get the interactive answer that we need. The simplest feature is a gray scale pixel with a value between 0 and 255 or between 0.0 and 1.0. A single gray pixel is generally not very interesting because we cannot detect anything from it. However, pixel features have a number of issues that must be addressed.

noise and history blending

For a gray pixel there is the problem of noise. Many camera sensors will not report the same value for a pixel from frame to frame even though the sensed image has not changed. The extent of this noise will vary from camera to camera. The most common solution to this noise is to average the pixel value over f frames. The problem with this average is that we must store f frames and with each new image I_i we throw the image I_{i-f} away and add the new frame I_i to our

history. We must also sum up each of the *f* values for each pixel and divide by *f* to get the desired average. As *f* increases the space required and the computation required at each pixel grows. This is a problem for an interactive system where speed is critical.

Instead of averages we use the history blend image *H(i,f)*. This is calculated as:

$$H(i, f) = \frac{1}{f} I_i + \left(1 - \frac{1}{f}\right) H(i-1, f)$$

The history image at frame *i* is a blend of the current frame image and the previous history image. The contribution of the current frame I_i is still *1/f* as in the average, but the remaining value comes from the previous history blend. The result is not a true average of *f* frames but can behave very much like one and with less computation and much less storage. The influence of previous frames falls off exponentially as they become distant in time. Much camera noise can be eliminated by using *H(i,2)* rather than I_i.

color and lighting

Many targets can discriminated based on color. This is a very common technique for detecting skin. (However, it works poorly for African skin). Color is also used for detecting specifically encoded objects in the image. For example the end of a baton might be painted bright red so that it is very easy to track, provided nobody wears a bright red shirt. Simple color detection uses RGB as the three features.

One of the problems that can arise with pixel features is lighting changes. For example, if a cloud moves in front of the sun the brightness of everything is diminished. This makes the R, G, and B values for a given pixel quite different even though the object being viewed has not changed. This is frequently resolved by converting RGB to HSV and throwing away V. Because the value is the brightness, that is the component that will change the most in the presence of lighting changes. Hue and saturation can be very robust features. A final technique is to use all of RGB and HSV with a learning algorithm that selects relevant features (forms of Naïve Bayes, some linear approaches, and decision trees). The learning algorithm keeps what actually works. The caveat is that the training data must include a wide sample of actual usage cases, not just the obvious ones.

An interactive camera system can use infrared light. The first reason is that it is invisible to the user. An infrared light can illuminate a dark scene without disclosing its presence in normal use. Infrared cameras can view a scene without being deceived by RGB light being projected into the scene. Living objects, like people, are generally brighter in infrared than furniture or books. The human retina is highly reflective of infrared light. Cheap CMOS cameras are actually quite sensitive to infrared and must be filtered to work correctly in RGB. Note that sunlight contains lots of infrared and can wash out the image, making it useless.

edges

The human visual system does most of its interpretation of objects by looking at edges. This is why we can identify a person from a line drawing as well as from a picture. Edge features have the property of being relatively impervious to changes in lighting. Given a pixel at *I(x,y*) we can measure its "edginess" by the formula:

$$E(x,y) = \left(I(x+1,y) - I(x,y)\right)^2 + \left(I(x,y+1) - I(x,y)\right)^2$$

This is actually the square of the magnitude of a gradient vector *G(x,y)*, whose formula is:

$$G(x,y) = \left[\left(I(x+1,y) - I(x,y)\right) \qquad \left(I(x,y+1) - I(x,y)\right)\right]$$

The gradient vector *G(x,y)* is an estimate of the normal vector of an edge passing through *I(x,y)*. The gradient vector will be used in some shape features.

Region features

It is rare, in interactive use, that a single pixel is of interest. We usually want features that cover a region of the image. The problem with region features is that their cost is not constant as with pixel features. Generally the cost of a region feature is linear in the number of pixels in the region. Another reason to consider whole regions is we may want to use a "skim and focus" technique to speed the interactive response time of our system. We first want to skim across an image with rather coarse region-based features to find likely places for more focused computing. For the skimming step we need very efficient region features that can be used to discard areas that are obviously irrelevant to our task.

For example, if we are looking for faces we can first train a classifier to look for skin using 10x10 region features. With this technique we need only look at every 5^{th} pixel in X and Y and thus look at $1/25^{th}$ the number of pixels. When we

find a region that shows "skin" we can then use pixel level features to work out whether it is a face or some other skin. This "skim and focus" technique is a very important speedup when using images for interaction.

One very effective tool is the integral image[9]. Integral images impose a one time cost on the entire image and then provide access to information about arbitrary sized regions in constant time. The definition of an integral image *INT(x,y)* is the sum of all of the pixels above and to the left *I(x,y)*.

$$INT(x,y) = \sum_{i \le x} \sum_{j \le y} I(i,j)$$

This integral can be computed in linear time starting at the upper left pixel and proceeding left to right, top to bottom. By proceeding in this order previous integrals will have already been computed and can be used efficiently.

$$INT(x,y) = I(x,y) + INT(x,y-1) + INT(x-1,y) - INT(x-1,y-1)$$

The final subtraction term is essential because those pixels will have been summed twice from the previous two terms.

Once we have the integral image we can calculate the sum of any rectangular region in constant time. Given a rectangular region whose lower right corner is at *(x,y)* with a width of *W* pixels and a height of *H* we can compute the sum of the pixels in the rectangle as

$$SUM(x,y,W,H) = \begin{array}{c} INT(x,y) - INT(x-W,y) - INT(x,y-H) \\ + \\ INT(x-W,y-H) \end{array}$$

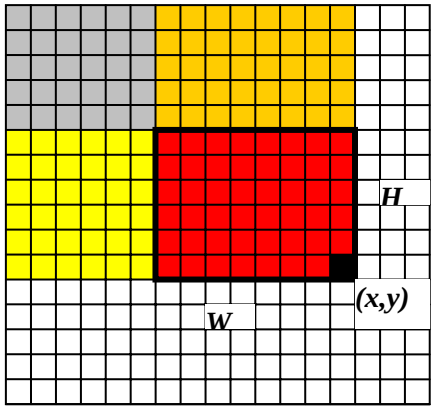
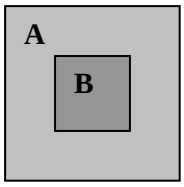


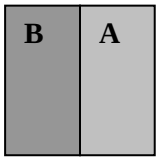
Figure 24.3 – Integral Images

A more useful value than the sum is the average (sum / area). Using the average function and an integral image we can use rectangles of various sizes to compute composite features such as shown in figure 24.4.

Spot = Ave(A)-Ave(B)



VertEdge = Ave(A)-Ave(B)



HorizEdge = Ave(A)-Ave(B)

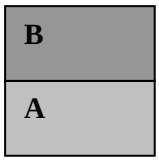


Figure 24.4 – Composite features from integral images

There are other integral images that can be formed using different base functions to be summed. For example we can integrate the squares of the pixel values using the formula:

$$INT2(x,y) = I(x,y)^2 + INT2(x,y-1) + INT2(x-1,y) - INT2(x-1,y-1)$$

Using the *INT2* integral image we can compute the sum of the squares of any rectangular region as:

$$SUM2(x,y,W,H) = \begin{array}{c} INT2(x,y) - INT2(x-W,y) - INT2(x,y-H) \\ + \\ INT2(x-W,y-H) \end{array}$$

With the *INT* and *INT2* images precomputed, we can compute the standard deviation of any rectangular area in constant time using the formula

$$STDV(x,y,W,H) = \sqrt{\frac{SUM2(x,y,W,H)}{W \cdot H} - \left(\frac{SUM(x,y,W,H)}{W \cdot H}\right)^2}$$

We can also measure the amount of "texture" in a rectangular region using the edge function *E(x,y)* to form an "edginess" integral image *INTE*.

$$INTE(x,y) = E(x,y) + INTE(x,y-1) + INTE(x-1,y) - INTE(x-1,y-1)$$

Shape recognition

In many cases it is shape information that identifies the item that we are looking for. The simplest shape technique is *template matching*. This is essentially a nearest neighbor classifier technique (Appendix A3.1c). We take a sample of what we are looking for and compare it with all possible pixel positions in the image looking for the closest match or the set of matches that are closer than some threshold. It is most common to use a rectangular region for the comparisons. Essentially we are transforming each pixel in the region into a feature in a feature vector (Appendix A3.1). We then use some distance metric to compare our template's features against the sample region of the image. The cosine distance (Appendix A1.1e) is a good choice. The normalization step in the cosine distance tends to diminish the differences due to lighting.

There are two major problems with template matching. The first is that it tends to fail in the presence of rotation. For example figure 24.5 shows an image

of a finger on the left with a finger search template on the right. The template will never match the finger even though they are both drawn from the same finger image. The only general solution is to use many templates each at a different rotation angle.

Figure 24.5 – Finger with finger search template

The second problem with template matching is the cost of the match. The image fragment and template in figure 24.5 are each 80x80 or 6400 pixels. This makes each match very expensive and it must be applied at every pixel in the image. This match cost can be improved using region features and the "skim and focus" technique described above.

A second shape technique is the use of image moments[10]. An image is first converted to gray scale where the shapes we are trying to identify are light and the background is dark. Using image moments we can compute a variety of features including the center of mass of the object, its width, height, rotation angle and many other features. See Appendix 1.3b for details on image moments.

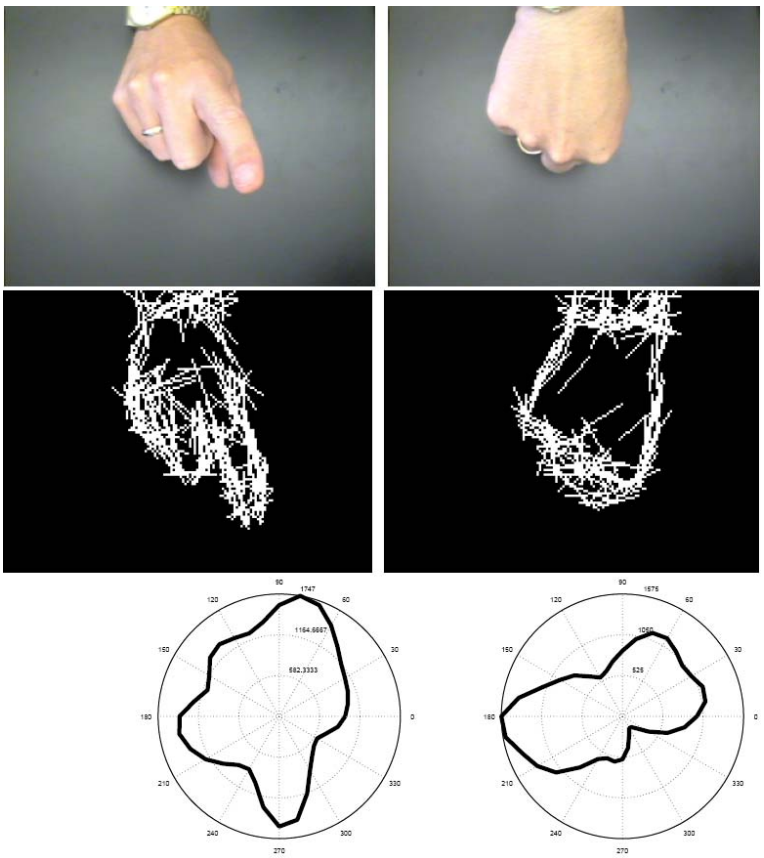


Figure 24.6 – Orientation Histograms[10]

A third shape feature technique is the *orientation histogram*[10]. This technique looks exclusively at edges because they are robust relative to lighting and also capture simple information about object shape. For each pixel *(x,y)* we compute the gradient vector *G(x,y)*. We discard any pixel for which the length of *G(x,y)* is below threshold as not being an edge. For the edges that remain we assign them to one of 8 to 16 "buckets" based on the orientation direction of *G(x,y)*. For a given image region we count the number of pixels that fall into each orientation bucket to produce a histogram of the edge orientations. This orientation histogram forms a feature vector for classifying shapes. Figure 24.6 shows an example of orientation histograms being used to recognize hand gestures. Note that the two gestures show a very different shape signature in the orientation histograms. Orientation histograms are also easier to work with when rotations

are present. A rotation in image space is just a shift in orientation histogram space. Thus with 12 buckets only 12 shifts and compares are required to capture all possible differences in rotation.

We can use integral image techniques to rapidly produce orientation histograms for any image region. If there are 12 orientation buckets then we compute 12 integrals that sum the number of pixels that match the orientation for a given integral. We can then get the sum of the pixel for each orientation for any rectangle in constant time. This can greatly speed the process of finding an object that matches a particular orientation histogram.

Motion features

A last set of features are those that detect interactive change or movement in an image. The simplest movement algorithm is *background differencing*. For this we take an image of the space in its quiescent state (no movement or interaction) and subtract it from the current image. Pixels with strong differences are where something has changed. For example we could take a picture of an empty office and then use current video images. If there are lots of changed pixels then there is probably someone in the office moving around.

Simple background differencing has two major problems. The first is camera noise. Simple noise will show up as pixel differences, which is inappropriate. The second problem is that the background is never static. If the sun goes behind a cloud then every pixel will change. If the user moves a chair and then leaves, that chair will be permanently different from the original background.

Both of these problems can be reduced by using history blending. We can use a short duration blend to eliminate the camera noise and a long blend to represent the background. The duration of the background blend depends upon the type of motion that we are trying to detect. If we want to detect the presence of a moving person we might us a background of *H(i,10*30)* which will blend across approximately 10 seconds at 30 frames per second. For foreground we might use *H(i,2)* to average out camera noise. The difference *H(i,2)-H(i,60)* will show basic body movement such as shifting in a chair or hand movement. Lighting changes as the sun goes down or clouds move will be too slow to cause much of a difference across 10 seconds. If we want to detect rapid motions then we might use *H(i,2)-H(i,10)*. The background is now changing at about 1/3 of a second and only very rapid changes will show up as strong differences.

Note that integral image techniques can be applied to the history blend images. Computing the blend of the integral image produces a blended integral

over time. This gives us the sum of the blends for any rectangle. Using these differences we can very rapidly search an image in large chunks for any movement that has occurred, without checking each pixel individually.

Background differencing can show where changes have occurred but will not give any indication of the direction of movement. Simple approximation of *optical flow*[10] can give us a useful indication of direction of motion. This model assumes that the object(s) being tracked are light and the background is dark. For this simple model of optical flow there are three cases that can be applied to a pixel $I_f(x,y)$.

1) $I_f(x,y) > I_{f-1}(x,y)$ In this case the pixel is growing lighter over time meaning that the object being tracked is entering the pixel. The direction of movement therefore is towards neighboring dark pixels and away from approaching light pixels. The movement vector for this pixel is towards the neighboring pixel [$I_f(x-1,y)$, $I_f(x+1,y)$, $I_f(x,y-1)$, $I_f(x,y+1)$] that has the lowest value.

2) $I_f(x,y) < I_{f-1}(x,y)$ In this case the pixel is growing darker over time meaning that the object being tracked is leaving the pixel. The direction of movement therefore is towards neighboring light pixels and away from dark pixels. The movement vector for this pixel is towards the neighboring pixel [$I_f(x-1,y)$, $I_f(x+1,y)$, $I_f(x,y-1)$, $I_f(x,y+1)$] that has the highest value.

3) $I_f(x,y) \approx I_{f-1}(x,y)$ In this case there is no detectable movement at that pixel.

The above steps can provide a movement direction estimate for a given pixel but such pixel sized estimates not reliabl. However, the average movement direction over a region can indicate a general direction of movement.

Summary of image features

The previous section has described features for gray values, pixels and regions of pixels. These are all quite easy to calculate. For shapes we have cosine matching of image templates, image moment features and orientation histograms. For movement we have background differencing and a crude optical flow estimate. These basic features combined with a fast classifier algorithm (Appendix A3) provide the foundation for many interactive techniques in physical space.

Training image classifiers

Many of the techniques describe below use some kind of a classifier either to locate an object in an image or to detect whether an object is present. Because of the prevalence of these tasks in camera-based interaction, Fails et. al. created Image Processing with Crayons[11]. In many of these cases the goal is to classify each pixel in an image as belonging to one of two or more classes. In Crayons each desired class is associated with a colored "crayon" that the user can use to paint the image. The crayons are semi transparent so that the user can see the image. In figure 24.7 the user is painting with a "skin" crayon and a "background" crayon in the left image. After painting some of the pixels the user requests the system to generate a classifier. The painted pixels are used as the training data. The resulting classification is shown in the image on the right as an additional transparent layer. The transparent layers help the user to understand how successful the classifier has been and where there are errors. The user can continue to paint corrections or move to other images to enter more data. The idea is that anyone who can understand how to paint can create a new image classifier within the limits of the feature set and learning algorithm.

Figure 24.7 – Image Processing with Crayons

Crayons uses 250 features generated from various sizes and combinations of integral image features. A decision tree (Appendix 3.1b) is used to train the classifier because it functions naturally as a feature selector. For a given problem only 5-10 of the 250 features are actually used. The decision tree algorithm selects those that are most useful. Because a Crayons user can easily produce tens to hundreds of thousands of training examples very quickly, the training time can become slow. The Crayons goal was to never take more than 10 seconds to train a classifier. In most cases only 1-2 seconds is required. Some modifications to the decision tree algorithm were required to achieve this goal.

Crayons are simple to build and is very effective at producing classifiers with little user effort. The right side image in figure 24.7 shows a problem that most Crayons users need to address. Classifying the interior of the hand is very different from classifying the edges of the hand. If knowing where the edges of the hand are is important to the task then more training data can be painted to improve the accuracy. However, training edges inherently "fights against" the training of interiors. Getting it all correct requires a rather sophisticated classifier. If what is desired is that only the location of the hand be identified, then continuing to train on the edges is frequently counter productive. Awareness of this edge/interior tradeoff can greatly simplify the construction of new classifiers to track objects in a camera image.

Camera as a pointing device

The first camera technique is to simulate a mouse or pointing device. The Direct Pointer[12] uses a cell phone camera and a wireless network link to control the position of a cursor on the display. A cursor is drawn on a screen and the user points the cell phone camera at the cursor. The camera grabs an image and transmits it to the cursor server. The curser server locates the cursor in the camera image and then moves the cursor on the screen so that it will be closer to the center of the camera image. This feedback loop pushes the cursor in the direction the camera is pointing. When the user moves the camera to the left the cursor will appear in the right of its image. The cursor server will then move it to the left to compensate. The feedback loop causes the cursor position to converge without knowing the optics of the camera. The advantage of this technique is that anyone with a cell phone can control a cursor on an arbitrarily large screen. Experiments with Direct Pointer showed a performance similar to a track ball.

Another cell-phone based technique is to place a camera above the display screen and use it to detect the display on a cell phone. The C-Blink[13] system puts a unique image on the cell phone and the display's camera detects where the image is. Moving the cell phone around will change where it appears in the display camera's image and this position can be used to control a cursor. C-Blink can also transmit button press information by flashing the cell phone's display. The display camera detects the flashes and can decode their timing into a binary value indicating what buttons have been pressed. Unlike the Direct Pointer, C-Blink requires much larger physical movements of the user to control pointer. C-Blink's most important contribution is in signaling information through the visual channel.

The most popular camera-based pointing device is the Nintendo Wii-remote. A sensor-bar is place on top of the television. This bar has 10 infrared LEDs with 5 in one end and 5 at the other. The Wii-remote has an IR camera that detects these two collections of LEDs. The average of their positions in the camera's image gives a cursor location. The angle of the vector between the two spots in the camera image gives an angle of rotation. Because the distance between the two ends of the sensor bar is known, the distance between the two points in the camera's image gives an estimate of the distance between the Wii-remote and the sensor bar. When the remote is closer the points will appear farther apart in the camera image. The use of infrared and known emitters makes the location of the spots a simple image processing task.

A final camera pointing technique uses a simple laser pointer as an input device[14]. The display is projected onto a surface and the user points at the display with a laser pointer. A camera focused on the display surface is used to detect the laser spot as shown in figure 24.8. The red arrow shows the laser spot, which is not as easy to detect as one might think. The blue arrow shows where the system believes the laser spot is located. The problem is the time lag of the image processing required to track the laser spot. The problem with detecting laser spots is that their brightness overdrives most cameras. This requires turning the camera gain way down so that only the spot is visible or introducing a very dark filter over the camera.

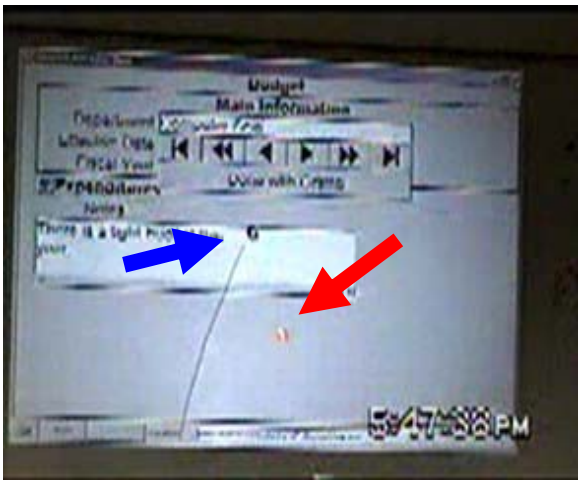


Figure 24.8 – Laser pointer interaction

One of the key reasons for laser pointer interaction is to allow many people to interact with a wall display rather than just one. Anyone with a laser pointer can participate. A variety of "laser widgets" were implemented, including dates, numbers, lists of choices and Graffiti text as shown in figure 24.8. Accuracy with the laser pointer is a problem. Myers et. al.[15] measured the effectiveness and accuracy of laser pointers and found that hand jitter was a serious issue.

When using a camera to input points there is the problem of calibration of the camera image to the input space. As figure 24.8 shows, the camera is frequently off center and viewing the surface at an oblique angle. The most appropriate place to mount a camera to minimize this distortion is also a place where the user's head is likely to be. Therefore the camera is usually in a location that has inherent distortion in the image. What we need is a mapping from camera image coordinates [Xc,Yc] to input coordinates [Xi,Yi]. A simple linear mapping is not sufficient because of the perspective and keystone distortions that can occur. The following mapping is sufficient to resolve most problems.

$$X_i = aX_c + bY_c + \frac{c}{X_c} + \frac{d}{Y_c} + eX_c^2 + fY_c^2$$

$$Y_i = gX_c + hY_c + \frac{j}{X_c} + \frac{k}{Y_c} + lX_c^2 + mY_c^2$$

The problem is to determine the coefficients *a* through *m*. There are 12 unknowns. If we have 16 points for which we know the mapping then we can use linear least squares (Appendix A1.2f) to approximate the coefficients. This is done by projecting 16 points onto the display surface one at a time and having the camera software find them in the image. We know in input space where we projected the point and we know in camera space where we found it. Doing this in a 4x4 grid of points will produce enough data for least squares approximation of the mapping.

Camera tracking of objects

A second role for a camera is to track objects in the physical world. The Tangible NURBS system used a rear projected wall display for its interactive surface. In addition to the projectors, infrared light is also shown on the back of the screen as well as cameras sampling the screen from behind. The user would interact with the system using plastic objects that had been modified so that they were visible in IR and their shape was distinctive. The distinctive shape allowed

the object to be identified as well as its orientation. The placement of the user's fingers also could be used as interactive input as shown in figure 24.9.

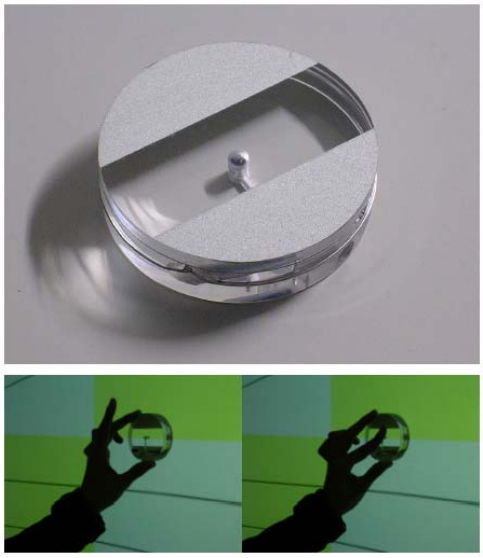


Figure 24.9 – Tangible NURBS shapes[16]

The Illuminating Light[17] project tracked objects on a table using cameras from above as well as projected images from above. The objects were identified by red, green and blue spots on the objects. These spots were arranged so that the combination of colors yielded the object's identity, location and orientation. As with the Tangible NURBS this system allowed users to interact physically with real objects rather than indirectly through a mouse and keyboard.

Frequently these objects being tracked by the camera are paper with information on them. Not only can the camera find the pieces of paper, but can also acquire images of what is drawn on the paper. The Designers' Outpost[18] focused on the need for web designers to rapidly modify designs using physical "sticky notes". The problem with sticky notes alone is that the interconnections between notes are lost when notes are moved. Also the physical notes could not be sent to remote coworkers. In Designers' Outpost there is a rear camera behind the screen with the projector. This camera can locate the adding, removing and repositioning of sticky notes on the screen. Because the camera is behind the screen, the user does not obscure this information. Interconnections are drawn digitally using a pen-input device. When sticky notes are moved the connections are also moved. Once the positions of sticky notes are known, a high-resolution,

but slow, camera extracts the information written on the sticky notes from the front. The user interacts by writing on notes, placing them on the screen and moving them around by hand. The camera system keeps the digital representation synchronized with what the user does with the physical objects.

Continuing in the theme of using physical objects to manipulate digital information, the PaperWindows[19] project uses pieces of paper as window controllers. Each piece of paper is augmented with 8 IR reflective markers. These are readily located by an overhead camera. Once the position and orientation of the paper is known, information can be projected from above onto the paper, as shown in figure 24.10. Moving a window of information around is now as simple as moving the corresponding piece of paper.

Figure 24.10 – PaperWindows[19]

Not only can the markers be detected in IR, but also the user's hands. A variety of interactive techniques are possible. Holding a piece of paper makes that window the active window. "Flipping" the top of a piece of paper will page through that document.

Camera tracking of paper can also be used to associate physical objects with their corresponding digital objects[20]. This system identifies rectangular objects that enter the camera's view. These can be documents or photographs. These might be printed versions of digital documents. For each rectangle a set of interesting texture points are identified in the image. The same is done for each digital object. The point pattern for the objects in the camera is matched against those in the database of digital objects to make the association. Using this

recognition, digital objects can be sorted into piles or rearranged in many ways. The physical actions with the objects are matched in the digital world.

Person Tracking

The last major camera-tracking approach is to watch people. In particular we watch their hands and fingers because those are the body parts most frequently used to manipulate the world. Letessier and Berand[21] describe an algorithm for tracking multiple fingertips at 25Hz which is more than enough for interactive use. They use background differencing and then apply a threshold to separate the foreground hands from the background. They then successively scan each pixel applying a series of filters to identify circular finger tips of appropriate width. Once we know where the finger tips are, there are a variety of interactive techniques that can use the information.

Malik and Laszlo[22] describe a tracking technique that uses the whole hand to find the fingers. They also use thresholded background differencing to find the moving hands. They then recognize that fingers are always connected to hands and the hands must enter the image somewhere. They use a flood-fill technique to find the two largest blobs in the thresholded image. If a blob is above a certain size then it is assumed to be a hand. They then look at the contour of each blob to find the sharp turns that indicate finger tips. By knowing where the fingers are attached to the hands they can recognize a number of gestures such as “grasping” projected objects and moving or rotating them. They then extended this work[23] to define a series of pointing, grasping, selecting and movement gestures for interacting with objects on a screen.

Most of the hand/finger tracking work uses a camera placed above the work surface. This has problems when users naturally bend over their work. There is also a problem if determining when the user’s finger touches the table top. Touch is an important part of detecting engagement with the work rather than just passing over it. The PlayAnywhere[24] system addresses both of these problems by using projectors, cameras and lights at an oblique angle to the surface. Using an NEC WT600 projector that has a very short throw distance, the projection is from in front of the user rather than above, which reduces occlusion. The camera and an IR illuminant are also placed at this oblique angle and quite separate from each other. This produces strong shadows of the hands in IR. Normally we try to eliminate shadows but in this case the shadows are used as the primary detection signal. The shadows have a uniform intensity regardless of the user’s hand color and are therefore more consistently detected.

The shadows also indicate touching the surface. When a finger is off of a surface, the shadow and the finger image separate because of the oblique angle of the light. When the finger touches, the shadow and the finger image converge with the finger obscuring the shadow. When detecting shadows this effect appears as a "sharpening" of the shadow finger tip as shown in figure 24.11. This shadow sharpening can be detected as a touch on the surface.

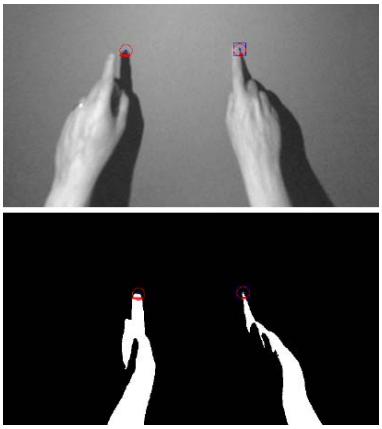


Figure 24.11 – Finger shadows to detect table touching[24]

An interesting technique puts the camera beneath the working surface. Acrylic plastic and glass exhibit the property of *total internal reflection*. That is when light is projected into the polished edge of a sheet of plastic it bounces off of its surface edges at an angle that keeps the light inside the sheet. When a finger is pressed against the surface it frustrates this total internal reflection causing light to reflect off of the finger and into the region behind the surface. A camera placed behind the surface can easily locate these touch points. By using an IR light source there is no glow from the surface at all.

The Light Widgets[26] system allows interaction designers to place cameras in a space and then create interactive controls by drawing them on the images. These controls are then sensitive to hands touching those places. Figure 24.12 shows views of a bedroom from two different cameras. A linear light widget has been drawn on the bedpost in each view (in yellow) and associated with the volume control of the room's television. A spot light widget (in red) has been drawn on the night stand and associated with the television's power switch. A skin classifier has been trained to detect when someone's hand has been place over one of the light widgets.

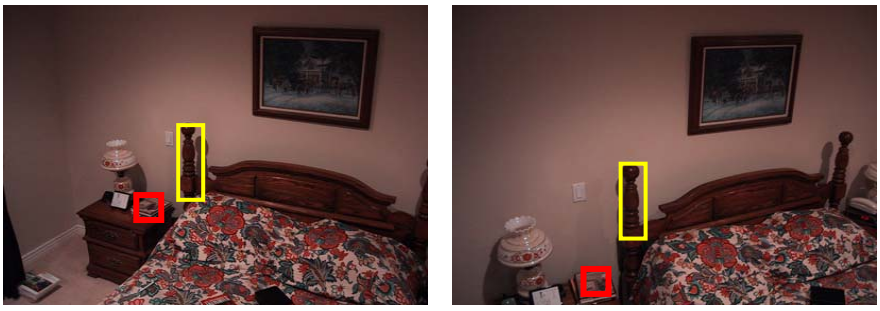


Figure 24.12 – Drawing a Light Widget

For spot widgets only the detection is reported from the camera. For linear widgets the proportional value (bottom to top) is reported. If the user's hand is actually on the desired spot then both cameras will report the same value. If the user's hand is above the spot, somewhere in the air, then the cameras will disagree. This simple agreement mechanism resolves 3D contact without expensive reconstruction of the 3D position of objects in the room. Light Widgets essentially allow a control to be drawn on any surface. A variation on this using laser spots rather than skin requires only one camera because a laser spot cannot appear in the air and the additional voting is not required.

One of the problems with putting cameras into living spaces is privacy. An internet-connected camera in the bedroom is not a good thing. By packaging the image processor with the camera, images need never leave the camera device. The device only reports light widget values that can be compared for agreement. Because no images leave the device, privacy is preserved.

A final class of techniques involves mounting the camera on top of a display and watching the user[27]. As the user waves their hands or makes gestures, the interpreted result is echoed on the display. This feedback approach sharply reduces the need for accurate calibration and recognition. The user can move their hand while watching the echo on the screen until the echo matches what is desired. This is very much like visually following a screen cursor while moving a mouse. The position of the mouse is largely irrelevant as long as the cursor reaches the desired location.

Researchers at MERL[10] have used cameras and on-screen feedback in a variety of computer gaming applications. Using image moments to detect the angle of hand gestures they could fly an avatar through a game. Using image

moments on body position a user could fly a magic carpet by leaning left or right. Magic spells are fired by pointing with the arms. Using orientation histograms allowed a user to play "paper, rock, scissors" with the computer. Using optical flow techniques to detect speed and direction of motion was used in a track and field video game. Speed of motion in the left and right of the screen detected the speed of the user running in place to win a foot race. Running, while lifting the arms to indicate a jump, supported a hurdle race.

Physical Sensing

Though the camera is a very versatile sensor there are other means for getting information about user activity. We will briefly look at several alternatives in this section.

Simple Sensors

A very common way to interact with an object is to touch it. Touch is easy to sense using body capacitance. A capacitor consists of two conductors separated by a dielectric. Various materials have various dielectric properties. The greater the dielectric, the more capacitance or ability the circuit has to hold a charge. Air is a rather poor dielectric. Water is a much better dielectric and the human body is mostly water. If some part of a human body establishes a circuit between two conductive surfaces, the capacitance between those surfaces increases sharply. This increase in capacitance can be easily sensed thus creating a touch sensor.

Sometimes squeezing an object has meaning. The conductive foam in which integrated circuits are packaged has a resistance that can be measured. When the foam is compressed the resistance lowers because there are more internal contacts within the cells of the foam. Inserting two wires into the foam some distance apart creates a pressure sensor that can be cut to any size or shape.

Proximity of an object to a sensor is easily measured in two ways. IR proximity sensors are relatively cheap. One side is an LED that shines infrared light into the environment. The other side is an IR light detector. The closer a reflective object is to the sensor the brighter the light on the detector. These generally do not work after a few feet but are quite effective at short range. IR sensors do not work if the target object is not reflective in the IR range or if a very bright IR source (such as the sun) is present in the environment.

Another cheap proximity sensor is ultrasonic and behaves much like sonar. The device emits an ultrasonic "chirp" and then times the echo. Short times mean short distances. Soft surfaces tend to absorb the sound reducing the effectiveness

and changes in humidity can create variations in the range. The most annoying attribute from a user interface perspective is that the ultrasonic "chirp" can frequently be heard as a faint audible click.

MEMS (Micro-Electro-Mechanical Systems) technology has created very cheap accelerometers. A single device will generally sense acceleration along two axes. A second device mounted perpendicular to the other can provide the third axis. These are very cheap and very small. Sensing the acceleration of gravity can create a tilt sensor. These devices generally produce a stream of acceleration values. Using appropriate features on these sensor streams coupled with a good learning algorithm can produce sensors for shaking, walking, running, tipping and holding still.

Sensor interaction

As an example, a telephone can have a conductive surface where the user holds the phone and a separate surface where the phone touches the ear. This allows the phone to sense when the user is actually holding the phone to their ear. Conductive paint can be used on any surface for this purpose. Dietz and Yerazunis[28] instrumented such a phone. When the phone is held to the user's ear it behaves normally. When the user takes the phone from their ear the phone starts to save the audio being received. When the phone is again placed to the user's ear the audio is played back at a higher speed allowing the user to catch up with the conversation.

A similar technology has been used to prototype a wide variety of touch sensitive controls[29]. In this case a user sits on a conductive pad to form one pole of the circuit. A processor has several touch sensitive surfaces wired to it. Each surface is tested in turn to see if there is a touch. Any surface that can be made conductive can sense touch and be used as a prototype device. In addition, multiple people can each sit on their own conductive pad. By sampling each circuit combination the system can not only sense a touch but identify the user that touched it based on which pad completes the circuit.

Hinckley[30] instrumented a handheld PDA with a touch sensitive case, an IR proximity sensor and accelerometers. The PDA knows when someone is holding it by the touch. It knows when it has been put into a pocket because the accelerometer indicates that the device is upright, the proximity is sensed as extremely close and there is no touch. The accelerometer can tell when the object has been tipped and will rotate the display to correspond to the edge that is currently down. The drawing application will erase the screen when the PDA is shaken while being held upside down.

Virtual Containers

There are a large number of techniques that can be classified as "virtual containers." A virtual container is a place on some server where data or a program is stored. Physical objects are given an identity that can be sensed and that identity is associated with the container. If the container holds a program or fragment of code, then that object represents the execution of that code to perform some service. If the object represents a data value then information can be put into the container or information can be used from the container whenever the associated object is present.

At the heart of this architecture is the ability to associate identifiers with physical objects. We will first look various techniques for digitally identifying objects and their associated interactive techniques.

Object properties

One of the simplest techniques is to use the physical properties of the object itself as its identity. Some researchers have used the weight of an object as its identifier[31]. A small scale is placed near a display device and the user places some personal object (cell phone, car keys, watch, etc) on the scale. The user then drags information on the screen to the location near the scale and the information is "dragged into the object". The user then picks up the object and carries it to a different display and places the object on the scale found there. The information "in" the object pops out onto the screen.

The association between the object and the information is based on the weight. When information is "dragged into" the object it is actually stored on a server indexed under the object's weight. When the object appears on a new scale the weight is looked up on the server and the associated information retrieved. Other potential properties are color or width/height.

The problem with most object-property techniques is that the property must remain invariant. The shape of a set of keys varies radically depending on how you set them down. A second problem is that the range of possible values that can be reliability detected is quite small. With the weight technique only 20-30 separate objects can be recognized.

Bar codes

A very common labeling technique is the bar-code. Most commercial products already have a bar code. However, UPC codes only identify the type of object, not the particular object. There are several advantages to bar-codes. The

first is that they are very easy to print and attach to objects. They are very cheap. A bar-code can also be designed to contain an IP address for the server that holds the object's virtual container. This allows such techniques to scale up to millions of codes with millions of differing applications managed by many organizations. The potential interactive techniques can scale to the size of the Internet.

There are two problems with bar-code techniques. The first is that they must be visible despite the fact that they are ugly. Most objects that people use in their environment do not have prominent bar-codes because they are not very esthetically pleasing. The second problem is the readers. Standard UPC codes are intended to be read by a laser scanner. The store-based scanners are too bulky for interactive use. There are handheld scanners, but they are generally about twice the size of a PDA. It is relatively easy to develop a bar-code scheme that can be read by a cell phone camera. The number if bits possible in the identifier is generally reduced, but small cameras attached to computer are much more widely available than laser-based readers.

Resistor IDs

The mediaBlocks[32] system used simple resistors as identifiers. Each block has a resister with differing resistance embedded into it. Around the room there were various devices with associated "transfer slots." Inside a transfer slot were two contacts that would engage with any block placed in the slot. Through the contacts the resistance was measured and the block identified.

The blocks were intended to "hold" media content such as images, video and slide shows. Dragging media materials to a transfer slot would put the media "into" the block. Inserting the block into a printer's slot would print the media on the block. Inserting the block into the slot associated with a camera would place any pictures or video taken by the camera "into" the block for use elsewhere. Each block becomes a physical embodiment of the information that it contains.

In addition to slots, the mediaBlocks system also had *racks*. A rack is basically a printed circuit board on which contacts of various lengths are etched. Putting a media block onto a rack at different positions causes the block to connect to different contacts. A processor can sample each set of contacts and identify each block and where it is on the rack. This position information allowed users to physically sort media information by adjusting the positions and ordering of various blocks on the rack.

There are two key limitations on resister IDs. As with object properties the range of possible identifiers are not very high. Additionally there must be

electrical contact for the block to be correctly sensed. The contacts must be kept clean or the resistance changes. In addition, the user must engage directly with the sensing device, unlike camera techniques.

iButtons

The iButton from Dallas Semiconductors is a small microcontroller with unique identifier encoded in it. This identifier has enough bits to accommodate Internet-sized systems. The microcontroller is encased in a metal shell that provides two contact surfaces. When the iButton is engaged with an appropriate reader, contact is made with the shell, power is supplied and the iButton can communicate its identifier. This has the same physical contact properties as resister IDs but has a much more robust range of identifiers. One of the down sides is that solid digital contact must be ensured so that digital information can be reliably transferred. This means that contact between an iButton and a reader is less like touching and more like snapping in place. This is not a physically satisfying solution.

A variation of the iButton has a Java interpreter embedded in it along with a small amount of persistent storage. This means that such a Java button can actually be the container for information without the need for access to a network server. The Join and Capture[33] system used an iButton embedded in a ring so that users could move their interactive work to whatever resources they found in the world. Each location was associated with either a display or some information.

RFID

An emerging technique for digitally labeling objects is RFID tags. The most interesting RFID tags are not powered and essentially static. They are very inexpensive. Inside the tag is a coil antenna. The tag reader contains a much larger antenna and induces a current in the tag's antenna. The induced current in the tag is sufficient to power the circuit that modulates a return signal with the tag's identifier. Identifiers up to 96 bits and beyond have been created. This is a great ID space. RFIDs do not need to be touched to be read. However, the reading distance is dependent upon the size and power applied to the reader's antenna. This means that readers are physically larger than one would like.

Want et. al.[34] have developed a series of prototypical uses for RFIDs in interactive systems. The essence of their idea is that objects should have an interactive meaning associated with what the object actually is. For example a tagged French-English dictionary can be brushed against a computer equipped with an RFID reader. Having detected the dictionary, any selected text is

translated from French to English. Brushing the computer against a poster that advertises the scheduling of a presentation will log that presentation into the user's calendar. Brushing the computer against a tagged book can locate that book on the Internet for purchase. If someone's business card has been tagged, brushing it against the reader causes that person's information to be entered into the contact manager.

The 96 bits of identifier information make RFIDs very flexible. The first 32 bits can contain the IP address of the server that stores the virtual containers. The remaining bits can identify the container. Each of the techniques described above either delivers some information or a piece of code to be executed. There are some ambiguity problems in these techniques, however. Brushing a book against the reader initiates a purchase request for that book. Brushing the dictionary initiates language translation. Why does it not initiate a purchase of the dictionary? A poster contains much more information than the time and place of the presentation. There are also the presenter and the topic. Why was the schedule determined as the associated meaning? With these ambiguities the users may be confused.

Fingerprints

One of the problems with associating information and actions with physical objects is that we tend to loose them. One solution is to associate containers with parts of the user's body. An interesting technique is to use one's fingers[35]. Each finger has a unique print that can be recognized by a reader. The technique is to scan the print and derive a set of features. These features are compared against all known information containers for the closest match. As with the other container systems the associated information and/or code can be used for a variety of interactive purposes.

The obvious limitation is that people generally have no more than 10 fingers. The toes could be used, but that would be awkward. The fingers also do not have the semantic associations that a dictionary or business card would have. This means that a user can easily become confused about the data/action associated with each finger.

Summary

Interacting in the physical world has the promise of integrating computing into our lives. To have interaction there must be the ability for the computer to present information. If computing is more integrated, then it must be more polite.

Computers that barge into our lives with noise and light at unwanted times will not be acceptable.

The second problem is to sense what the user is doing or what the user wants done. One of the most flexible sensors is the camera. For interactive use we do not necessary need or want all of image processing. A few simple features and techniques can generate the information we need from which interactive behaviors can be trained using machine learning.

In addition to the camera there are a variety of other sensors including proximity, pressure, touch, orientation and acceleration. Again machine learning can be used to translate these inputs into interactive behaviors.

A compelling form of interaction is to associate physical objects with information or behaviors. The key requirement is a technology for identifying the object. A most important question is whether the identifying technology can scale to millions of uses and whether the users will understand what the associated behavior is.

---

[1] Weiser, M., "Some Computer Science Issues in Ubiquitous Computing", *Communications of the ACM 36(7)*, ACM (July 1993), pp. 75-84.

[2] Underkoffler, J. and Ishii, H. "Illuminating Light: An Optical Design Tool with a Luminous-Tangible Interface", *Human Factors in Computing Systems (CHI '98),* ACM (1998), pp. 542-549.

[3] Raskar, R., Welch, G., Cutts, M., Lake, A., Stesin, L., and Fuchs, H., "The Office of the Future: a Unified Approach to Image-based Modeling and Spatially Immersive Displays", *Computer Graphics and Interactive Techniques (SIGGRAPH '98)*, ACM (1998), pp. 179-188.

[4] Summet, J. W., Flagg, M., Rehg, J. M., Abowd, G. D., and Weston, N. "GVU-PROCAMS: Enabling Novel Projected Interfaces", *ACM Multimedia*, ACM (2006).

[5] Ishii, H. and Ullmer, B., "Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms", *Human Factors in Computing Systems (CHI '97),* ACM (1997), pp. 234-241.

[6] Heiner, J. M., Hudson, S. E., and Tanaka, K. "The Information Percolator: Ambient Information Display in a Decorative Object", *User Interface Software and Technology (UIST '99),* ACM (1999), pp. 141-148.

[7] Dickie, C., Vertegaal, R., Sohn, C., and Cheng, D., "eyeLook: Using Attention to Facilitate Mobile Media Consumption", *User Interface Software and Technology (UIST '05),* ACM (2005), pp. 103-106.

[8] Fogarty, J., Hudson, S. E., Atkeson, C. G., Avrahami, D., Forlizzi, J., Kiesler, S., Lee, J. C., and Yang, J., "Predicting Human Interruptibility with Sensors", *ACM Transactions on Computer-Human Interaction 12(1)*, ACM (Mar 2005), pp. 119-146.

[9] Viola, P. and Jones, M. "Robust real-time object detection." *Technical Report 2001/01*, Compaq CRL, February 2001.

[10] Freeman, W.T., Anderson, D., Beardsley, P., Dodge, C., Kage, H., Kyuma, K., Miyake, Y., Roth, M., Tanaka, K., Weissman, C., and Yerazunis, W., "Computer Vision for Interactive Computer Graphics," *IEEE Computer Graphics and Applications*, IEEE, (May-June 1998), pp 42-53.

[11] Fails, J. A., and Olsen, D. R., "A Design Tool for Camera-based Interaction," *Human Factors in Computing Systems (CHI '03)*, ACM (2003), pp 449-456.

[12] Jiang, H., Ofek, E., Moraveji, N., and Shi, Y., "Direct Pointer: Direct Manipulation for Large-Display Interaction using Handheld Cameras," *Human Factors in Computing Systems (CHI '06)*, ACM (2006), pp 1107-1110.

[13] Miyaoku, K., Higashino, S., Tonomura, Y. "C-Blink: A Hue-Difference-Based Light Signal Marker for Large Screen Interaction via Any Mobile Terminal", *User Interface Software and Technology (UIST '04),* ACM (2004), pp. 147-156.

[14] Olsen, D. R., and Nielsen, T. "Laser pointer Interaction," *Human Factors in Computing Systems (CHI '01)*, ACM (2001), 17-22.

[15] Myers, B. A., Bhatnagar, R., Nichols, J., Peck, C. H., Miller, R., and Long, C., "Interacting at a Distance: Measuring the Performance of Laser Pointers and Other Devices" *Human Factors in Computing Systems (CHI '02)*, ACM (2002), pp 33-40.

[16] Bae, S. H., Kobayashi, T., Kijima, R., and Kim, W. W., "Tangible NURBS-curve Manipulation Techniques Using Graspable Handles on a Large

Display", *User Interface Software and Technology (UIST '04),* ACM (2004), pp. 81-90.

[17] Underkoffler, J., and Ishii, H., "Illuminating Light: an Optical Design Tool with a Luminous-tangible Interface" *Human Factors in Computing Systems (CHI '98)*, ACM (1998), pp 542-549.

[18] Klemmer, S. R., Newman, M. W., Farrell, R., Bilezikjian, M., and Landay, J. A., "The Designers' Outpost: A Tangible Interface for Collaborative Web Site Design", *User Interface Software and Technology (UIST '01),* ACM (2001), pp. 1-10.

[19] Holman, D., Vertegaal, R., Altosaar, M., Troje, N., and Johns, D., "PaperWindows: Interaction Techniques for Digital Paper," *Human Factors in Computing Systems (CHI '05)*, ACM (2005), pp 591-599.

[20] Kim, J., Seitz, S. M., and Agrawala, M., "Video-Based Document Tracking: Unifying Your Physical and Electronic Desktops", *User Interface Software and Technology (UIST '04),* ACM (2004), pp. 99-107.

[21] Letessier, J., and Berard, F., "Visual Tracking of Bare Fingers for Interactive Surfaces", *User Interface Software and Technology (UIST '04),* ACM (2004), pp. 119-122.

[22] Malik, S., and Laszlo, J., "Visual Touchpad: A Two-handed Gestural Input Device" *International Conference on Multimodal Interfaces (ICMI '04)*, ACM (2004), pp. 289-296.

[23] Malik, S., Ranjan, A., and Balakrishnan, R., "Interacting with Large Displays from a Distance with Vision-Tracked Multi-Finger Gestural Input", *User Interface Software and Technology (UIST '05),* ACM (2005), pp. 43-52.

[24] Wilson, A., "PlayAnywhere: A Compact Interactive Tabletop Projection-Vision System", *User Interface Software and Technology (UIST '05),* ACM (2005), pp. 83-92.

[25] Han, J. Y., "Low-Cost Multi-Touch Sensing through Frustrated Total Internal Reflection", *User Interface Software and Technology (UIST '05),* ACM (2005), pp. 115-118.

[26] Fails, J. A., and Olsen, D. R., "Light Widgets: Interacting in Everyday Spaces," *Intelligent User Interfaces (IUI '02)*, ACM (2002), pp. 63-69.

[27] Eisenstien, J., and Mackay, W. E., "Interacting with Communication Appliances: An Evaluation of Two Computer Vision-based Selection

Techniques," *Human Factors in Computing Systems (CHI '06)*, ACM (2006), pp 1111-1114.

[28] Dietz, P. H., and Yerazunis, W. S., "Real-time Audio Buffering for Telephone Applications", *User Interface Software and Technology (UIST '01),* ACM (2001), pp. 193-194.

[29] Dietz, P. H., Harsham, B., Forlines, C., Leigh, D., Yerazunis, W., Shipman, S., Schmidt-Nielsen, B., and Ryall, K., "DT Controls: Adding Identity to Physical Interfaces", *User Interface Software and Technology (UIST '05),* ACM (2005), pp. 245-252.

[30] Hinckley, K., Pierce, J., Sinclair, M., and Horvitz, E., "Sensing Techniques for Mobile Interaction", *User Interface Software and Technology (UIST '00),* ACM (2000), pp. 91-100.

[31] Streitz, N. A., Geibler, J., Holmer, T., Konomi, S., Muller-Tomfelde, C., Reischl, W., Rexroth, P., Seitz, P. and Steinmetz, R., "i-LAND: An Interactive Landscape for Creativity and Innovation," *Human Factors in Computing Systems (CHI '99)*, ACM (1999), pp 120-127.

[32] Ullmer, B., Ishii, H., and Glas, D., "mediaBlocks: Physical Containers, Transports and Controls for Online Media," *Computer Graphics and Interactive Techniques (SIGGRAPH '98),* ACM (1998), pp 379-386.

[33] Olsen, D. R., Nielsen, S. T., and Parslow, D., "Join and Capture: A Model for Nomadic Interaction," *User Interface Software and Technology (UIST '01)*, ACM (2001), pp 131-140.

[34] Want, R., Fishkin, K. P., Gujar, A., and Harrison, B. L., "Bridging Physical and Virtual Worlds with Electronic Tags", *Human Factors in Computing Systems (CHI '99)*, ACM (1999), pp 370-377.

[35] Sugiura, A., and Koseki, Y., "A User Interface Using Fingerprint Recognition – Holding Commands and Data Objects on Fingers," *User Interface Software and Technology (UIST '98)*, ACM (1998), pp 71-79.