TOWARDS RELIABLE COMPUTER VISION-BASED TANGIBLE USER INTERFACES

Audrey J.W. Mbogho
Computer Science Dept.
City University of New York Graduate Center
365 Fifth Ave, New York, NY 10016
USA
ambogho@gc.cuny.edu

Lori L. Scarlatos
Dept. of Computer and Information Science
Brooklyn College, CUNY
2900 Bedford Avenue, Brooklyn, NY 11210
USA
lori@sci.brooklyn.cuny.edu

ABSTRACT

A major obstacle in the development of computer vision-based interfaces is the uncertainty in the image data. Guaranteeing reliable program behavior while the inputs cannot be relied upon to take any specific values is desirable but extremely challenging. We have experimented with various strategies for addressing this problem in a controlled environment and have identified some that we find promising. Using visual tags, we encode object attributes, capture them with a camera, and read them. The test applications are implemented as Macromedia Director movies with Lingo scripting, while the image analysis module is implemented in C++ as a Lingo Xtra. We conclude that evaluation through rigorous experimentation and testing is a suitable approach for discovering techniques that work well and the conditions that maximize the chances for optimal performance.

KEYWORDS

Visual tags, tangible user interfaces, reliability.

1. Introduction

The history of computing shows a clear gradual continuum [1],[2]. We have moved from the machine centered interfaces of the early days, requiring the user to engage in a great deal of manual activity, to the graphical user interfaces that currently dominate. More recently, gaze (e.g. [3]) and gesture recognizers hint at a future with fully human-centered, perceptual interfaces. Among the most promising advances in this area are the tangible user interfaces, which blur the distinctions between data and its representations [4]. With these interfaces, users manipulate data by manipulating physical objects in a very natural way. In order for the computer to respond appropriately, it must be able to track the physical objects in the environment. This is often achieved with either sensors or computer vision. One of the problems with tangible user interfaces is that there are no standardized input devices; each application dictates what the interface should look and feel like.

We have addressed this problem by developing a suite of visual tagging strategies, and a library of functions for detecting those tags. Using our strategy, any ordinary object – such as a puzzle piece, math manipulative, or science specimen – can be made part of a tangible user interface.

At the moment, full-fledged, sensor-based interfaces face major obstacles. Some experimental ones include cumbersome headgear, gloves, or other contraptions that must be worn and manipulated in specific ways in order for users to communicate their intentions to the computer. Sensor systems are also limited in terms of the number of sensors that may be tracked simultaneously, and therefore do not scale easily.

Interfaces based on computer vision are attractive in that the sensor (i.e. the camera) can receive input passively, without discomfort to the user. Cameras can also track large numbers of objects with minimal impact to the cost or complexity of the software. In addition, because of falling prices, cameras are readily available and are now being integrated into computing devices, such as, cellular phones. The main problem that has to be overcome is the uncertainty inherent in image data. Noise – the random effect resulting from changing environmental conditions such as lighting, temperature, pressure, dust, humidity, motion, and so on – causes the same image point to be represented differently from frame to frame. Furthermore, the sampling rate, which is dictated by camera technology, and occlusion both result in loss of information. In [2] the observation is made that general purpose computer vision is still a far away goal, and that to achieve real-time robust vision-based interfaces, we must constrain problem domains, exploiting properties specific to each.

An image identification algorithm, therefore, typically makes certain assumptions about its inputs. As a result of the uncertainty in the data, however, images may fail to conform to algorithmic assumptions, causing the algorithm to give inaccurate results. Errors in a vision system that assists a pilot or a surgeon can result in loss of life. Errors in a vision system assisting children in a classroom can cause the children to become frustrated and impede rather that facilitate learning. It is this latter scenario that this research focuses on: how to develop accurate vision-enabled interfaces in educational settings and how to measure this accuracy.

Bearing in mind that a given vision-enabled interface (VEI) cannot deliver the same degree of reliability all the time, we need to be able to state how a system will perform under a given set of circumstances. In addition, within the ideal circumstances where a given VEI has the potential for optimal performance, we need to determine the set of input parameters that will make this desirable performance a reality. In other words, we recognize two situations that can make an otherwise good VEI perform poorly. One is when it is put to use in a problem domain for which it is not suited. The other is when input parameters are poorly chosen. The goal of this research is to address both these issues through experimentation and evaluation of educational environments augmented with visual tags.

2. Visual Tags

If the environment in which a vision system operates can be controlled, we can overcome the problem of imperfect image data by emphasizing the things that we want to detect. One thing that we can do is identify those things that we want to track with visual tags. A tag can be a plain marker carrying one bit of information (present or absent), or it can incorporate various shapes and colors encoding a binary string of as many bits as the application requires. By tagging, the problem of handling the large amount of data associated with a large and complex image is reduced to one of finding a small object (tag) which stands out in its environment and whose appearance is known a priori. A good tagging scheme should maximize the amount of information it can accurately hold in a small area.

In [5], TRIP, a system that uses barcodes and cameras to locate users and objects in living and working environments is described. One proposed application is the use of a video camera to record locations of tagged books in a library and link a book's information to the electronic catalog database with its exact physical location in the stacks. In another application a software agent associated with each person locates that person and migrates an audio player and MP3 decoder to the computer nearest to him. A third use is in recognizing when a user is near a workstation and then logging that user on automatically.

CyberCode, described in [6], is a tagging scheme for use in augmented reality environments. The tag is a 2D black and white barcode that is captured using low-cost CMOS or CCD cameras such as those that come attached to mobile devices. As with TRIP, both the tag and its location are identified. One use is putting the tags on printed documents to provide a link between the hard copy and its electronic version. In another use, the tag, once captured, brings up menu items on the screen that can then be selected by moving the tagged object around.

In both [5] and [6], the barcodes are black and white. The use of color is often shunned because of the difficulties it presents. These difficulties arise from the wide range of shades a color can take as illumination changes. That is, the problem of uncertainty is worse in the presence of color. On the other hand, color offers additional evidence for or against an object's identity. Humans use color this way all the time, and it would be helpful if machines could have the same ability. Therefore we made the choice to use color in our applications and attempt to reap the benefits it does offer over black and white or grayscale analysis. Specifically, we used the YIQ color model, which allows one to analyze and manipulate luminance and chrominance information separately.

Color analysis is similarly used in [7], but with the RGB model, to segment the hand in a system which replaces the mouse with a pointing finger.

The system proposed in [8] is similar to ours in its use of color and tangible objects as inputs. The physical objects are color cubes that a user arranges on a semi-transparent surface. A webcam under this surface captures this arrangement, which represents a sketch of an image the user wants to retrieve from a database. This "query by example" approach tries to match the appearance and placement of colors in the sketch with images in the database.

We are still experimenting to discover colors that stand out best in the environment and that are easily distinguishable from one another. We have experimented with three tag designs, which we describe next, mostly focusing on the color barcode as it best illustrates the problem of imperfect data.

Unlike TRIP and CyberCode, our tags have a simple design. This simplicity allows for fast image analysis, which is crucial for true real-time interactive computing.

The classroom setting is an ideal test-bed for visual tags. Many educational activities involve the identification of objects. For example, playing physical puzzles involves recognition of the puzzle pieces. Tagging the pieces simplifies the automation of this task. Similarly, abstract concepts can be represented using drawings on physical objects (e.g. wooden cubes) which can then be tagged so that the computer can recognize them as well. The basic idea is to have a computer quickly recognize an object, and perhaps also its position relative to other objects, so that a vision-based application can guide, correct, or encourage a child attempting to also identify that object as part of an educational exercise. For each activity, the tagging scheme chosen must allow the representation of enough values to cover all the different objects used.

2.1 Single Bit (SB) Tag

The SB tag is identified by its color and the presence or absence of a black patch, about 1/3 the size of the tag, at the center (see figure 1). For each color, the SB tag represents a 0 if it has a patch and a 1 if the patch is absent. We read the SB tag by computing the proportion of the tag's area that is not covered by the black patch. If this proportion is close to one, we conclude that there is no patch and set the tag value to 1. Otherwise, the tag value is set to 0. SB tags can be used to represent attributes that have two values, such as, "yes" or "no." For example, a classroom activity might require children to separate seashells according to whether they are hinged or unhinged. SB tags can be used for computer vision-based assistance in this activity.

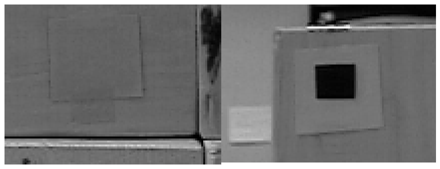

Figure 1: An orange SB tag with value 1 and a green SB tag with value 0.

2.2 Punch Hole (PH) Tag

In PH tags, shown in Figure 2, each tag has a different number of holes punched into it. A tag with no holes represents the value zero. A tag with n holes represents the number n. A 2" by 1" tag can accommodate up to 15 holes, permitting the representation of 16 values per color. To read PH tags, we first calculate the area occupied by a tag with no holes. For tags with holes in them, we count the number of pixels of the tag color still remaining on the tag. Thus the largest area represents 0, while the smallest area represents 15, for a given color.

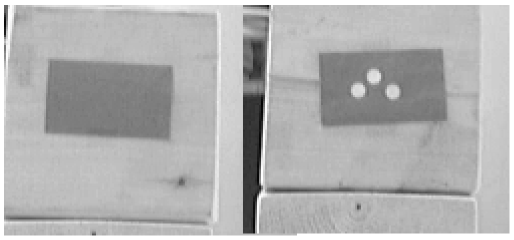

Figure 2: PH tags with values 0 and 3.

2.3 The Tricolor Barcode

Other barcode-enhanced user interfaces have been proposed, but many are black and white and require a special barcode reading device. For example, in [9], a bookmark management system is introduced for barcoding objects so that they can be linked to the web addresses with which they are associated. Scanning the barcode using a barcode reader causes the current web page address to be stored in a database. Scanning while holding down the Ctrl key brings up the addresses previously linked to the object.

Our tricolor barcode consists of three different colored tags, such as, red, green, and blue. On a tag is a barcode made up of black and white bars. The leftmost bar is white. It marks the beginning of the code and also indicates what color represents binary 1. Black bars represent binary zero. The code itself begins at the second bar and is read up to the middle of the tag. The remaining half of the tag is a mirror image of the first half, so that the code can be read in either direction. Figure 3 shows an example of a tag representing binary 2.

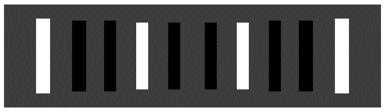

Figure 3: Barcode representing binary 2 (0010)

2.4 Dichotomous Sorting

The tricolor barcode will initially be used in a dichotomous sorting classroom activity for children in the 6 to 8 age range.

Dichotomous sorting involves asking children to take miscellaneous objects and sort them according to some predefined category. Tags such as the ones shown in figure 1 are affixed to each object. Each tag color represents a category or attribute, such as, rock density, origin, type and so on (see figure 4, for example).

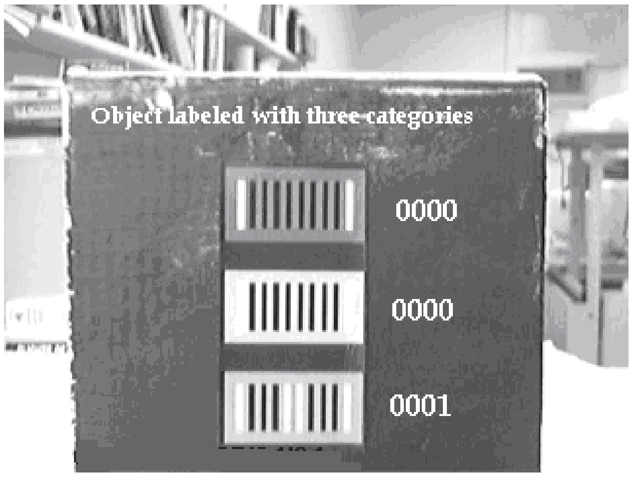

Figure 4: Each barcode color represents a specific attribute. The number encoded by the barcode represents the value of that attribute for this object.

The children place the objects on a mounted glass surface with a camera below [10] (see Figure 5). Under the glass is a camera that continuously sends a live video stream to the computer running the dichotomous sorting application. When the application receives a request to process the current image, a sample is grabbed off the stream and analyzed in a manner, described below, that ensures a tag can be read regardless of its orientation.

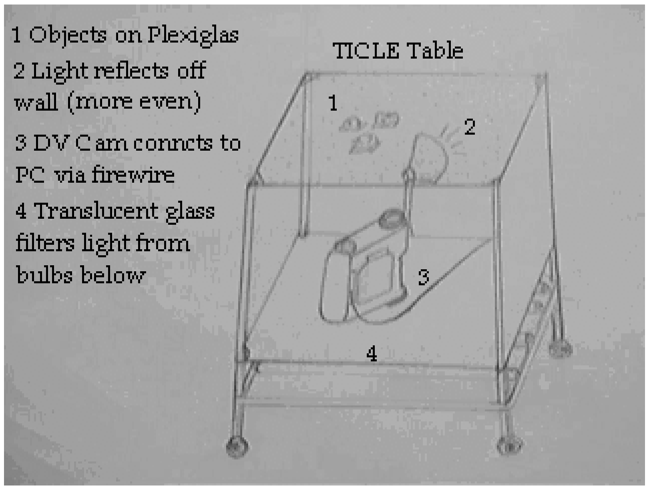
1 Objects on Plexiglas
2 Light reflects off wall (more even)
3 DV Cam conncts to PC via firewire
4 Translucent glass filters light from bulbs below
TICLE Table
1
2
3
4

Figure 5: Prototype setup for dichotomous sorting classroom activity.

2.5 Barcode reading algorithm

Assuming for the sake of clarity that we are looking at red tags in the captured image, we first find the bounding boxes for all red tags in the image. These are encoded in a list, which is then saved to a file. Each 4-tuple in the list represents the minimum row, maximum row, minimum column, and maximum column values for a bounding box. Tags are read by analyzing the colors and color transitions inside the area enclosed by each bounding box. The tags may appear in the image oriented in any direction. The tag reading algorithm, based on the barcode reader found in [11], takes this into account by following the steps below.

1. Attempt to read the tag horizontally. If the tag is valid (i.e. has 20 transitions), go to step 5.
2. Attempt to read vertically. If the tag is valid go to step 5.
3. Attempt to read in a diagonal direction. If the tag is valid go to step 5.
4. Attempt to read the other diagonal direction.
5. If the final count of transitions is less than 20, output an error code; otherwise output the value read in one of the preceding steps.

3. Dealing with Imperfect Data

Initially, the system does not know what a red, green, or blue tag is. We have to train it to recognize these colors. We do this by taking a few pictures of the tags at the beginning. We click on a tag of each color multiple times. We have found that about 36 to 40 clicks are sufficient. The average YIQ triple is computed. To compute the luminance tolerance, we find the maximum absolute Y difference between all samples and the average. We compute the chrominance tolerance by finding the maximum square difference between the (I, Q) pairs of the samples and that of the average. We use this "farthest from midpoint" heuristic in order to avoid missing any pixels of a given color, but outliers can skew it. In future work we will try other heuristics.

We use the contour tracing algorithm, due to Pavlidis [12], to find tags. Contour tracing is sensitive to discontinuities that inevitably appear in the image due to the error prone nature of image acquisition cited earlier. As a result, we often find a much larger number of tags of a given color than are actually present. A partial solution to this problem lies in the fact that many of these are clearly invalid tags consisting of one pixel, a short straight line, or a small rectangle. The tag reading procedure will eliminate these, as it will not find the right number of bars in any of them. We consider this a partial solution for two reasons. The image can be of such poor quality that it is split into many small tags, none of which are valid. Secondly, when contour tracing produces a large number of tags, the application slows down noticeably. It is important to overcome this obstacle in our application, as children can easily grow impatient and abandon the system if it is slack to respond. Choosing the right tolerance values is crucial to solving this problem.

Another method that we have used to come up with suitable luminance and chrominance tolerance values is through manual inspection of images. We look at the image of a tag and if we see a wide variation in brightness over the tag, we use a larger value for luminance tolerance. Similarly if we see a wide variation in the color of the tag, we use a larger value for the chrominance tolerance. We have found that this technique results in tags being read correctly for a significant proportion of the trials. However, it is not immediately clear how to automate this process. In future work we will investigate an AI approach that somehow incorporates our knowledge of tag image appearance and the physical environment.

4. Experiments

We tested the barcode application by reading tags repeatedly. We varied the experiment in four ways: (1) using a web cam with a direct view of the tags, (2) using a web cam viewing tags through Plexiglas, (3) using a digital video camera with direct view, and (4) using a digital video camera through Plexiglas. Ideally, for a single tag, only one bounding box should be found. This was the case when the application was tested with perfect synthetic images. Multiple bounding boxes are found in real images when degraded quality results in color discontinuities. Thus the number of bounding boxes found per tag is a good indicator of accuracy: the smaller this number, the more accurate the system. The proportion of trials in which tags are read correctly provides additional evidence with respect to accuracy.

	Bounding boxes per tag	% Correct readings
Webcam - direct	1	100
DVC - direct	1	100
DVC - Plexiglas	20	50

Table 1: SB Tag identification and reading accuracy

	Bounding boxes per tag	% Correct readings
Webcam - direct	1	50
Webcam -Plexiglas	1.18	72
DVC - direct	1	62.5

Table 2: PH Tag identification and reading accuracy

	Bounding boxes per tag	% Correct readings
Webcam - direct	1.5	90
DVC - Plexiglas	2.36	10

Table 3: Tricolor barcode identification and reading accuracy

The results, summarized in tables 1-3 above, show that, while the use of a Plexiglas surface is more convenient, direct view of objects affords greater accuracy. For applications that use schemes such as SB tags, in which pixel values are aggregated, degraded image quality arising from the use of Plexiglas is not a significant obstacle to accurate readings. However, where a single misrepresented pixel can break the application, for example in reading the color barcode, Plexiglas is not suitable. We have tried to use clear glass and found that it is worse than Plexiglas as it reflects objects from below and this obscures the objects of interest that are on top. Our current setup, however, does not allow for objects to be viewed directly; they need to rest on top of something transparent. We continue to work on discovering a setup that is convenient for the children to use and that provides high quality images at the same time. One of the ideas we will be testing is an upright structure with shelves on which to place objects from one side so that a camera on the opposite side can see them directly.

A rather surprising observation we have made is that the digital video camera images are of a lower quality, from a computational analysis point of view, than those obtained from a low-cost, low-resolution webcam.

We note also that the results for PH tags are inconclusive, with unacceptably high percentages of incorrect readings in all test cases. This is probably as a result of poor tag design or a poor reading method for this tag type. In future work we will look at the issue of tag design and tag reading techniques that suit particular designs.

5. Conclusions

We have implemented a library of functions that detects three different types of visual tags. This work contributes to the state-of-the-art by providing a generic way of tracking tangible user interfaces in an environment.

Our barcode reading algorithm works perfectly on perfect synthetic images; these barcodes are read correctly the first time. This underscores the importance of the problem of imperfect data in vision-enabled interfaces, especially when the intricate details of the image have meaning. We feel that this problem has not received enough attention in the literature. For any given new vision-enabled interface, there will be conditions that can easily arise where it can fail completely. We need greater acknowledgement of this fact and increased research into how to deal with it. While it may not be possible to produce systems that perform well under all circumstances, it would be very beneficial if we could enable users to determine the circumstances and input parameters that can give the more accurate performance. Our preliminary anecdotal observations indicate that this is possible through rigorous empirical evaluation.

6. Acknowledgments

This work is being supported by the National Science Foundation under Grant No. 0203333.

References:

[1] F. Quek, Eyes in the interface, *Image and Vision Computing, 13*(6), 1995, 511-525.

[2] F. Quek, Non-verbal vision-based interfaces. Keynote speech, *International Workshop in Human Computer Interface Technology,* Aizuwakamatsu, Fukushima, Japan, 1995.

[3] J. J. Magee, M.R. Scott, B.N. Weber, & M. Betke, EyeKeys: A real-time vision interface based on gaze detection from a low-grade video camera, *Proc. IEEE Workshop on Real-Time Vision for Human-Computer Interaction,* Washington, D.C., 2004, 159-166.

[4] B. Ullmer & H. Ishii, Emerging frameworks for tangible user interfaces, *IBM Systems Journal, 39*(3&4), 2000, 915-931.

[5] D. L. de Ipina, P. Mendonça, & A. Hopper, TRIP: A low-cost vision-based location system for ubiquitous computing, *Personal and Ubiquitous Computing Journal, 6*(3), 2002, 206-219.

[6] J. Rekimoto & Y. Ayatsuka, CyberCode: Designing Augmented Reality Environments with Visual Tags, *Proc. ACM Conf. on Designing Augmented Reality Environments*, Elsinore, Denmark, 2000, 1-10.

[7] F. Quek, T. Mysliwiec, & M. Zhao, FingerMouse: A

freehand pointing interface, in *Proc. International Workshop on Automatic Face and Gesture Recognition*, Zurich, Switzerland, 1995, 372-377.

[8] K. Matovic, T. Psik, & I. Wagner, Tangible image query, *Lecture Notes in Computer Science, 3031*, 2004, 31-42.

[9] P. Ljungstrand, J. Redstrom, & L.E. Holmquist, Webstickers: Using Physical Tokens to Access, Manage and Share Bookmarks to the Web, *Proc. ACM Conf. on Designing Augmented Reality Environments,* Elsinore, Denmark, 2000, 23-31.

[10] L. Scarlatos, TICLE: Using multimedia multimodal guidance to enhance learning, *Information Sciences 140*(1-2), 2002, 85-103.

[11] J.R. Parker, *Practical Computer Vision Using C* (New York: John Wiley & Sons, 1994).

[12] T. Pavlidis, *Algorithms for Graphics and Image Processing* (Rockville, Maryland: Computer Science Press, 1982).