HOP: Hands-On Projection
Final Report

A Report Submitted in Partial Fulfillment
of the Requirements for SYDE 462

Al Amir-khalili, 20198393
Chris Best, 20200501
Kyle Morrison, 20203836

Faculty of Engineering
Department of Systems Design Engineering

April 5, 2010.
Course Instructor: Dr. Dan Stashuk

Contents

List of Tables

List of Figures

1

Introduction

Computer technology has revolutionized the way people approach design and organization. However, it has also to a large extent taken physical interaction out of the design work flow. For tasks which are driven by spatial metaphors and benefit from direct tactile and visual feedback, the traditional approach of using a keyboard and a mouse as input devices leaves room for improvement. This project aims to address this need by establishing a system which combines virtual design and physical object manipulation. The goal is to create a system in which physical markers which are arranged on a workspace act as handles with which the logical virtual design can be created and manipulated. The ease with which many users can quickly visualize and communicate their ideas will be increased. The focal point of the design is allowing people to use natural spatial concepts and actions to interact with the computer design environment.

1.1 Problem Statement

Current visual design systems impose restrictions, whether physical or virtual, which create further separation between the designer and the design. Other systems, which attempt to bridge the gap between physical and virtual design, are cumbersome and restrictive. What is needed is a simple and intuitive method for integrating physical design with virtual models.

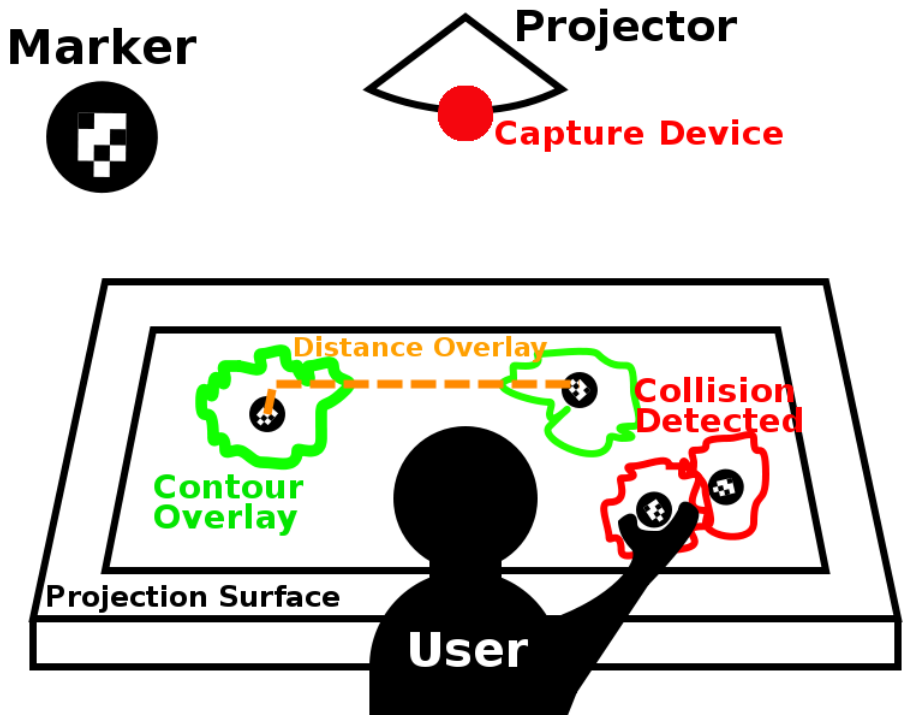


Figure 1.1: Hypothetical Layout of the System

1.2 Background

Humans have an innate aptitude for manipulating objects with their hands, and performing visual and spatial planning in the real world. For one, human beings are born with the ability to move, and by five months have started to move objects with their hands [1]. This is in stark contrast to computers, which must be learned after a maturation period, and which some people learn only after struggling [2]. By virtualizing the full appearance of the model while maintaining its tactile nature, a designer will be able to get the best of both worlds.

Current methods for a physical-virtual hybrid design system do currently exist. One of the first Augmented Reality systems, called CyberCode [3], used the same basic idea: use 2D pattern recognition to project a virtualized model into the real world. However, the CyberCode project only projected Computer generated graphics, meaning that only when viewed through a computer did the post-processed image

contain any extra information. To the people in the room at the time, the patterns they were looking at remained unchanged.

The idea of CyberCode has been extended on several occasions. One such extension was the Harbour Game [4], which was implemented via the ARToolkit [5]. However, the implementation method again precludes the designers from projecting the model into the real world, and instead they must rely on a computer generated model via a viewing screen.

Another such example is an Augmented Reality Workbench from MIT graduate students [6]. It incorporates physical manipulation with digital virtualization for the sole purpose of urban planning. There are several drawbacks with their system, however. The first is that it restricts the design domain to urban planning, and is thus inflexible. The second is that it requires a luminous table to project the design, which adds another barrier between the designer and the design. Finally, it requires the designer to build stick scaffoldings of buildings instead of using digital projections, adding more work for the designer.

2

Objectives

The primary objective is to create a functioning prototype of a spatial visualization and manipulation system. A representation model is desired which effectively encapsulates the information required for spatial planning. An input scheme which allows the user to modify their design by intuitively manipulating physical markers is designed, and a physical display will be built which clearly and accurately outputs the representation model while integrating the marker input system. To this end, the following technical objectives are used:

- **Design a Marker:** The group will design a class of markers which are comfortable and intuitive for a person to manipulate, which can represent a variety of spatial planning tasks, and which can be easily recognized using computer vision for the input to the system. This will involve researching the state of the art in marker design from an image recognition perspective, developing requirements from a user perspective, and combining these into a simple, robust design.

- **Implement a Recognition Algorithm:** The group will implement a computer vision algorithm which takes an image with the above markers and captures the spatial information they represent. This will involve researching how to design or adapt such an algorithm given the other constraints imposed by the model. In particular user interference like hands and arms temporarily ob-

scuring markers will need to be accounted for. Also, the algorithm will need to compensate for the effects of projecting things onto the markers themselves. The algorithm(s) used will be optimized so that real time feeling performance can be achieved.

- **Design a Visual Overlay Model:** The group will design an overlay which renders additional information from the model, such as dimensions, conflicts, and visual representations for display to the user. It will be necessary to research and determine what information is useful for the chosen application, and how to display it most clearly. For example, a floor planning model might show a warning when a fire exit was obstructed. In a battle planning scenario, it could show line of sight. The visual overlay will be take input in forms which are translatable from the output of the marker recognition subsystem.

- **Integrate Into a Physical System:** A surface will be designed on which the markers can be manipulated with augmented display information projected onto it in real time. Integrating all these components will give the overall effect of a virtual display which has physical handles by which it can be manipulated.

Concerns

This concept deliberately ties the manipulation of the model to physical objects. This also introduces limitations - something like scrolling would not be as straightforward to accomplish when the items in the scene are "connected" to the physical markers. This is a trade-off that will have to be handled carefully so that the resulting user experience is not crippled. However, the speed and simplicity of physical manipulation of objects compensates for some of the possible difficulties.

Another concern is the generality of the design. Even if it is as general as possible for spatial design tasks, there are still non-spatial but important design tasks (say, text input) which may be tightly coupled. It is acceptable for the project's scope to exclude non-spatial parts of the work flow, but there should at least be some thought given to how the project would fit into a full design process.

3

Methodology

3.1 Marker Detection

In the context of this project, Marker Detection refers to the technique used to determine the location of the control markers on the input plane. Given a frame captured from the input camera, which sees the input surface complete with markers, projected display, and any obstructions, it is necessary to extract usable position data for the markers. This allows the system to use the locations and movement of the markers as input. Accurate position information is also a precondition for bit extraction, as described in the next section.

Criteria

There are three main criteria on which marker detection algorithm will be evaluated.

1. **Accuracy** - The algorithm should recognize the position of each marker as accurately possible. This is measured by the deviation, in millimeters, of the (x,y) coordinates and radius r of each marker circle measured by the algorithm from the marker's actual position on the input surface. This is important both because the accuracy of the input to the application is fundamentally limited by the performance on this criterion, and because the position information is used in the bit extraction phase.

2. **Robustness** - The algorithm should be as robust as possible under the required operating conditions. This is measured jointly by the rate of false positives and false negatives for detection on an array of test images with varied conditions.

3. **Speed** - Because of the real time nature of the input system, it is important that the detection step introduce a minimum of latency. This criterion is measured by the worst case time, in milliseconds, that the detection algorithm takes to run.

Research

A great deal of the relevant work on geometric shape detection in images discusses the concept of Hough transforms, or an improvement thereof, as the core method for detecting circles or other geometries [7][8][9][10][11][12][13][14]. This technique is powerful in that it is quite general and robust, but it also suffers from performance problems [11]. This makes it well suited for problems where the end goal is to find any circle in an image, regardless of whether the circle is deliberately placed or naturally occurring.

Other approaches depart completely from the Hough approach, in most cases in order to gain greater performance at the expense of loss of generality. An example of this is [15] which uses gradient vector pairs, for a cost saving of eight times over a Hough method. At the other end of the spectrum [16], uses a genetic algorithm to achieve greater accuracy, but reports detection times of up to 5 seconds on a natural image. Finally, a very intriguing method called Inscribed Angle Variance was presented in [17] which is actually not doing image recognition at all, but rather doing mapping with laser range finders for a robotics application. The appealing part of this method is that it uses simple geometric properties of a circle in a very computationally efficient way, and achieves detection times on the order of milliseconds. Again, this approach trades away generality in that a) it sets a hard limit to the circle parameters it will accept, and b) it assumes a certain quality of input information (the laser's statistical range tolerance is known) but the approach excels at its intended purpose.

The way the Hough transform family of geometry detection methods work, at a high level, is to define a parameter space for the geometry, and to create an accumulator in the parameter space. The image (or a processed version thereof, usually with edge detection) is then processed in pieces, and evidence is gathered for whether the geometry exists at a point in the given parameter space. At the end, the totals are counted and a "vote" of some type in the parameter space is used to determine where geometry is found [14].

This has the advantage of being very general, but is computationally relatively expensive [11]. This means that in applications which require only specific (rather than general) detection - for example with known intensity level properties - but which also have strict time complexity constraints, Hough transform based methods are not ideal. In fact, if the application is specific enough, the generality of the approach becomes a drawback rather than a benefit, as shapes which are circles, but which are not the specific circles being searched for, are more likely to be detected. One approach to address this issue is to add a second classifier such as a neural net to prune the Hough circle results, as in [12]. However, this fix can only add to the time complexity problems.

Implementation and Results

Recall that the original reason for the investigation of circle detection techniques is to detect a specific type of circular marker, shown in Figure 3.1. In the image, the "Unknown" section refers to the space where data bits are stored, and thus should not affect detection. Experiments were conducted with OpenCV's Hough circle detection algorithm. These showed that the algorithm was actually sufficiently fast, with the time to process a frame on the order of 20 milliseconds. It also can successfully pick out the circular markers, as shown in Figure 3.2a. However, given a complex natural image, the false positive rate is enormous and wildly varying, as seen in Figure 3.2b. In live video form, it can be seen that not only are a great number of non-markers are detected, but where it detects these circles varies erratically between frames.

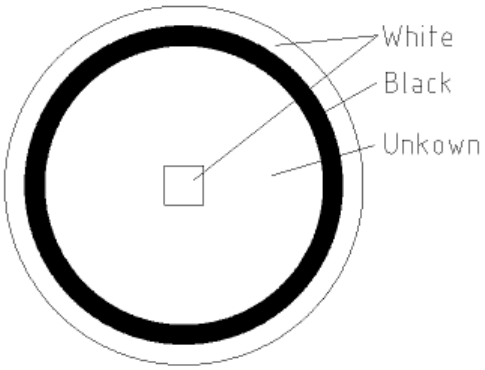
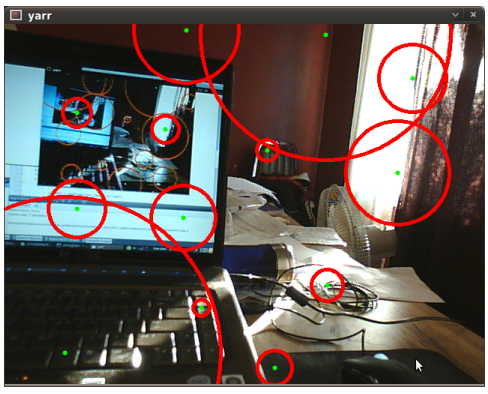


Figure 3.1: The Marker Specification

In order to fix this problem, an algorithm was devised with the requirement that it be as fast or faster than OpenCV's Hough algorithm, but which detects only the specific marker circles required for the application. This was accomplished by starting at the opposite end of the problem from where a Hough transform would start. Instead of sweeping the spatial domain in order to populate an accumulator in the parameter domain, the new algorithm sweeps the parameter domain, checking for a match in the spatial domain at each iteration.

(a) Marker in a simple natural image



(b) Complex natural image: many false positives are found

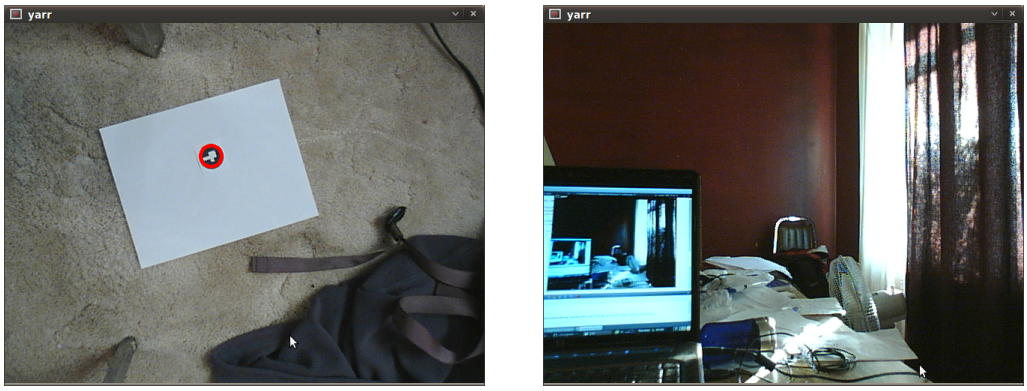
Figure 3.2: OpenCV's Hough Circle Detection

Define the parameters for the circle to be x and y the coordinates of the top point of the circle, and r, its radius. Then simply perform a grid sweep of interesting values for these parameters. At each stage, a boolean decision must be reached as to whether a marker is in the given location L = (x,y,r). Technically this gives us a complexity of M*N*R, where M and N are the number of horizontal and vertical steps in the image at the search resolution, and R is the set of all radius values necessary to check. However, because the algorithm is looking for a specific image, the boolean check at each stage can be short circuited, so that for the vast majority of iterations very little computation is performed. For example it is possible to define intensity value thresholds for what constitutes "black" and "white" in an image, and use the algorithm to decide if there is a marker in a given location.

Algorithm 1 Psuedo-code to match a marker

```
if the top pixel is not black then
  return false
end if
if the center pixel is not white then
  return false
end if
for all θ do
  if pixel at (r, θ) is not black then
    return false
  end if
  if pixel at (r + δ, θ) is not white then
    return false
  end if
end for
return true
```

In almost the entire image, the first check will fail, and no additional calculations are necessary. As an additional optimization, if the very top pixel fails, it is not necessary to sweep the radii at all, since by the definition resented here, no circle has its top at that location. Additionally, once a marker is found, it is possible to block out the entire space covered by that marker from future detection, as in the given scheme no markers may overlap.



(a) Marker in a simple natural image

(b) Complex natural image: no false positives are found

Figure 3.3: Parameter Sweep Circle Detection

The success of this algorithm is seen in figures 3.3a and 3.3b. Because of the parametric approach taken with the specific marker in mind, there are no false positives in the complex image, while still accurately matching the marker circle in the simple test image.

Test Results

Two tests were devised to evaluate the proposed detection algorithm against the criteria. The first is a robustness test, in which 140 markers are placed in a scene, and frame captures are taken under an array of sub-optimal lighting conditions. This captures are passed through the algorithm, and it is manually verified that there are no false positives. The error rate is then calculated as the number of markers not detected divided by the total. The time for each detection is also recorded, and this can be regarded as a worst-case time because both the aggravating conditions - a large number of markers, and bad lighting conditions - are present.

It can be seen from the tabulated robustness results in Table 3.1 that in most of the test images there is 100% detection, and in the three of twenty that have failures, the error rate is 2, 3, and 2 out of 140 respectively. Upon visual inspection, it is verified that these failures are caused by defects in the image caused by extremely

Table 3.1: Robustness Test Results

Image	Time (ms)	Miss Rate
00.ppm	28	0.000000
01.ppm	26	0.000000
02.ppm	26	0.000000
03.ppm	27	0.000000
04.ppm	26	0.000000
05.ppm	26	0.000000
06.ppm	26	0.014286
07.ppm	26	0.000000
08.ppm	25	0.000000
09.ppm	25	0.000000
10.ppm	25	0.000000
11.ppm	26	0.014286
12.ppm	25	0.021429
13.ppm	25	0.000000
14.ppm	25	0.000000
15.ppm	25	0.000000
16.ppm	24	0.000000
17.ppm	24	0.000000
18.ppm	25	0.000000
19.ppm	25	0.000000

poor lighting. It can also be seen that all detection times are under 30ms, which is well within the requirements.

The second test is an accuracy test, in which 6 markers were placed in a scene, and again 20 images in varying poor lighting conditions were captured. For these images, the true values of the circle parameters topX, topY, and radius were calculated manually to the pixel. The detection algorithm was then run, and the mean squared error for each of these parameters computed. The detection time was also measured. The results of this can be seen in Table 3.2. The mean squared error is less than 1 pixel in nearly all cases, and is at worst 1.3. Again, the detection times are very acceptable.

Table 3.2: Accuracy Test Results

Image	Time(ms)	Succeeded	Failed	MSE x	MSE y	MSE r
img00.ppm	13	6	0	0.000000	0.333333	0.333333
img01.ppm	12	6	0	0.333333	0.166667	0.500000
img02.ppm	24	6	0	0.333333	0.166667	0.500000
img03.ppm	19	6	0	0.666667	0.000000	0.666667
img04.ppm	19	6	0	0.333333	0.166667	0.500000
img05.ppm	12	6	0	0.833333	1.000000	0.500000
img06.ppm	12	6	0	0.000000	0.500000	0.500000
img07.ppm	12	6	0	0.000000	0.333333	0.500000
img08.ppm	12	6	0	0.166667	1.333333	1.000000
img09.ppm	14	6	0	0.000000	0.666667	0.833333
img10.ppm	25	6	0	0.166667	0.166667	0.666667
img11.ppm	14	6	0	0.000000	0.500000	0.500000
img12.ppm	20	6	0	0.500000	0.333333	0.666667
img13.ppm	29	6	0	0.166667	0.833333	0.833333
img14.ppm	31	6	0	0.166667	0.333333	0.500000
img15.ppm	18	6	0	0.000000	0.833333	0.833333
img16.ppm	14	6	0	0.166667	0.166667	0.333333
img17.ppm	20	6	0	0.333333	0.166667	0.333333
img18.ppm	13	6	0	0.500000	0.000000	0.333333
img19.ppm	19	6	0	0.666667	0.000000	0.500000

Thresholding and Flip Detection

Although the testing described in the previous subsection shows that the basic detection algorithm is performant and robust under the test conditions, there are areas where there is room for integration with the overall system.

The first is that when the top down projection is added to various other poor lighting conditions, the input image from the camera can be very degraded. In particular, the case of purely fluorescent lighting, combined with a DLP projector, results in a very poor captured image. This can be mitigated by applying adaptive thresholding to the image, as in Figure 3.4.

Figure 3.4: And adaptive thresholding filter removes artifacts from fluorescent lighting, and other lighting noise

Additionally, this method leaves room for tweaking to adjust for specific lighting conditions. The adaptive thresholding algorithm bins pixels into black and white depending if their intensity value is greater or less than the mean value in a neighbourhood around them. OpenCV's adaptive thresholding algorithm takes an optional parameter which allows for an offset to be added to this mean first however, which allows for fine grained adjustment of the thresholding behaviour with respect to mean intensity.

Another way the location detection can be improved is by fine tuning the center point and radius detection still further, as this directly impact bit extraction performance. The original algorithm is quite accurate, but there is a slight bias towards the left for circle centers, because of the scan direction. This is addressed by running the scan twice, once on the image as usual, and once on the image flipped both horizontally and vertically. The center and radius can then be averaged between the two, removing the directional bias and making the radius detection more accurate and more stable.

3.2 Bit Extraction

The process of bit extraction occurs after the markers have been located, and is integral to determining the identity of individual markers. It can be split into two processes: determining the orientation of the marker, and extracting the information bits from the marker. However, in the current design of the markers, the significance of each information bit is dependent on its location relative to the orientation bit, and thus these two processes must be done sequentially with the orientation identification coming first. The problem can be stated as: given the top and center of a circle, find the bit in the marker which determines its orientation.

Orientation Detection

Several methods are considered for finding the orientation bit, and thus determining the orientation of the marker. The first feature that jumps out about the orientation bit is that it always represents the farthest cluster of white from the center of the marker. Furthermore, it is a square, and thus has corners. These two facts suggest that one might be able to find the bit by simply finding the corners that are farthest from the center, or finding the clusters of white that are farthest from the center. Both of these approaches will be considered.

The first approach that comes to mind is finding the two corners farthest from the center of the marker. The first step is to use a corner detection algorithm to determine the parts of the marker that have the highest probability of being corners [18]. Then, step through these corners and keep track of which two are the furthest away from the center. Once the two corners have been identified, they can be used to create some basis vector $\mathbf{n}$, and the vector orthogonal to $\mathbf{n}$ the orientation vector $\mathbf{o}$. And thus the problem of finding the orientation has been solved.

The major problem with this approach is that it is very sensitive to loss of resolution. For a fully cropped, high resolution image of a marker this algorithm does quite well. However, as the marker gets farther away from the camera, and the resolution of the marker decreases, the image starts to lose some of the finer detail around the

corners. This can result in only one of the corners being found, or none being found at all. In the case of only one being found, the algorithm would be forced to look for another corner, and would instead find one of the information bits. This would lead to a disastrously erroneous estimate of orientation. One idea might be to only take the single farthest corner from the center instead of the two farthest. This would also prove difficult, and the resultant vector would be systematically skewed by a nontrivial angle, and the algorithm has no way of knowing which way the skew is and thus no way of correcting it. Even worse is the case where the algorithm does not find any corners. At this point, the behaviour of the algorithm for that particular mark would be difficult to predict, and even more difficult to correct.

Alas, for the flaws just outlined with the corner detection algorithm, the cluster detection algorithm is also not suitable for this particular problem. This algorithm can be described as attempting to find the farthest clusters of white pixels in the circle, and using the two farthest clusters as the corners of the orientation bit. As previously mentioned, however, this method is heavily reliant on the resolution of the image and the finer detail in the corner of the orientation bits. This observation is borne out in the testing. When the camera is close to the marker, the algorithm does quite well and identifies the orientation bit around 95% of the time. However, as the camera is moved farther and farther back, the performance of the algorithm becomes worse and worse. When the distance of the camera is sufficient to simulate real world applications (such as tabletop interaction), the detection of orientation can be seen to oscillate wildly over the marker.

Based on the poor performance of the aforementioned algorithms, a new approach was required which would not exhibit such drastic performance loss at lower resolutions. Of course, at a certain lower bound of resolution any algorithm is certain to fail, simply because there is not enough detail in the image. However, what was needed was an algorithm that would perform well within the distance measurements that are being considered for the application (between 0.5 and 1 meters). For the new approach, a third feature of the marker is considered which was previously ignored: the orientation bit is always a fixed distance from the center of the marker. Thus,

once the center of the marker is identified, the new process is to sweep radially around the center for possible locations of the orientation bit. Since the exact size in pixels of the marker is expected to change, the radius of interest is instead expressed as a fraction of the radius of the marker. In this case, the center of the orientation bit is expected to be at $\frac{5}{7} * r$ from the center of the marker. To further ensure accuracy, instead of the looking for the white pixels at this radius, the algorithm instead searches for regions of white pixels. A champion region (and an associated angle) is maintained by the algorithm, which updates only if the new region is, on average, white than the previous champion. Pseudocode for this algorithm can be seen below. This champion region method ensures that the edge of the orientation bit is never detected, only something that is close to the center of the orientation bit. This further reduces the complexity of orientation finding, because there is no longer any vector subtraction that was needed with the previous corner methods.

Algorithm 2 Psuedo-code to determine orientation

```
r ← 5/7 marker.radius()
for all θ do
  if pixel.localAverage() at (r, θ) ≥ currentChampion then
    bestTheta ← θ
    currentChampion ← pixel.localAverage()
  end if
end for
return bestTheta
```

This method works quite well in practice. Once a marker location is detected and passed to the algorithm, it is able to determine the correct orientation with a high degree of accuracy, as can be seen in Figure 3.5. The exact performance of the algorithm will be considered in the testing. These tests are not as rigorous as the full application testing that will follow after the calibration phase of the project, but there are enough to show the potential of the algorithm and the accuracy that it is capable of.

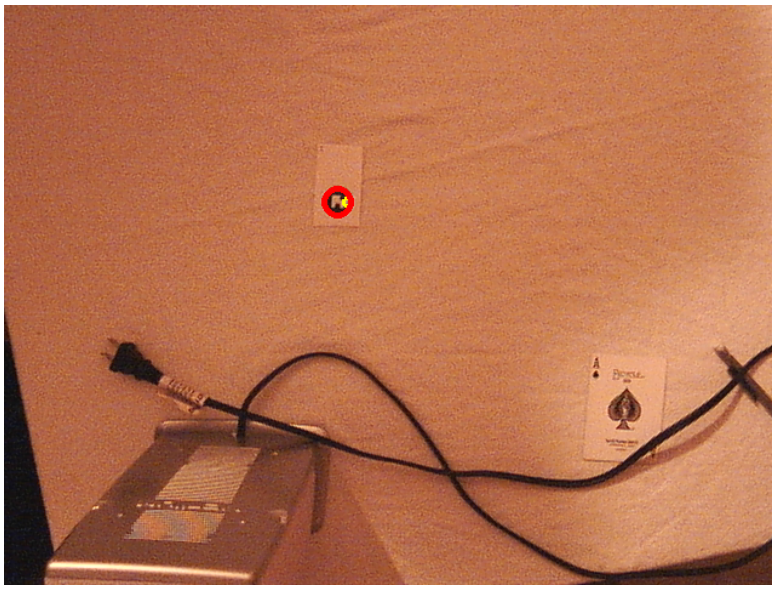


Figure 3.5: The marker is detected, and the orientation bit is correctly found

Information Extraction

Once the orientation of the marker has been determined, the actual information extraction is relatively trivial. Each of the 8 information bits can be uniquely identified by their expected radius to center and their rotation with respect to the orientation bit. The estimated location of the bits based on the orientation can be seen in Figure 3.6. The local average of pixels is taking at these 8 rotations and radii, and this average is compared against some threshold to determine whether the area is deemed black or white. This part of the algorithm is smooth, with no defects seen so far in the testing. These bits are then transformed into a binary representation, which is in turn transformed into some decimal number. With eight bits of information, it is possible to represent 256 distinct markers, which is more than enough for the purposes of this application.

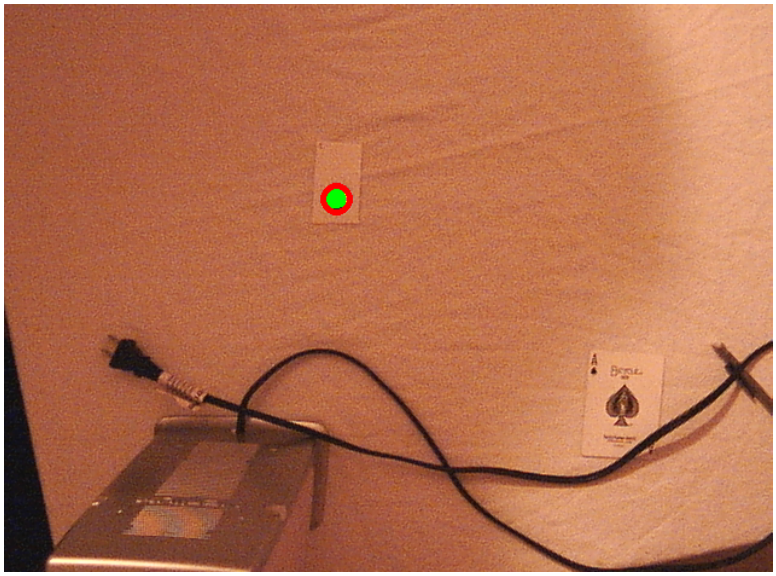


Figure 3.6: Using the orientation bit, the information grid is projected onto the marker

Testing and Evaluation

The testing of the bit extraction depends heavily on the accuracy of the marker detection as well as the resolution of the image. A steep drop-off in accuracy is noticed as the resolution of the markers decrease. As can be seen in Table 3.3, the algorithm more often that not gets more right than wrong. However, it is also clear that there is a large room for improvement, especially in the largely shaded images where (3 and 17) where the bit extraction gets more wrong than right.

One possible fix for these accuracy issues is making the markers slightly larger, and thus increasing the resolution of each marker. That will ensure more pixels (say 12 as opposed to 6) make up the orientation bit, which would allow greater accuracy in its detection. Another idea is to use a separate image for the bit extraction, and perform adaptive thresholding on it. This could eliminate the problems seen with shaded images, and would make the algorithm more robust.

Table 3.3: Accuracy of Bit Extraction

Image	Succeeded	Failed
img00.ppm	4	2
img01.ppm	6	0
img02.ppm	2	4
img03.ppm	6	0
img04.ppm	4	2
img05.ppm	5	1
img06.ppm	4	2
img07.ppm	4	2
img08.ppm	3	3
img09.ppm	4	2
img10.ppm	5	1
img11.ppm	4	2
img12.ppm	4	2
img13.ppm	4	2
img14.ppm	6	0
img15.ppm	6	0
img16.ppm	6	0
img17.ppm	1	5
img18.ppm	5	1
img19.ppm	2	4

3.3 Calibration

It is established that the resolution of the camera is going to be different from the output of the projector. Furthermore, it would be inefficient to design the system in such a way that the camera captures nothing other than the projected surface. In order to fill the gap between points detected in the camera space and where these points appear in the projection space, a calibration algorithm is required.

Homography

Homography, also known as collinearation, is an invertible transformation which maps the real projective planes onto a common projective plane. Such transformations are composed of pairs of perspective transformation. So, on a two dimensional plane, at least three points are required to form a three-by-three homography matrix [19].

3-Point Homography

In the first attempt, a simple 3-point homography technique developed by Fang and Chang was used [20]. This technique seemed to rely heavily on accurate detection of the three points. If a point is detected incorrectly, the geometric correction will heavily favour the incorrect direction. This effect is magnified further by the fact that the resolution of the capture space is less than half of the projection space. Therefore, if the detection is off by two pixels, it is effectively off by more than four pixels in the projection space. A better method is required which takes advantage of redundant homologous points (more than three).

5-Point Homography

A more robust method was proposed in Fang and Chang uses a 5-point pattern. This method may also be extended to use an arbitrary amount of points to achieve better accuracy. However, these algorithms were developed for calibrating touch-screen sensors. Touch screens tend not to suffer from pitch, yaw, and fish-eye distortions that are common with camera lenses. A better method is required which is unique to applications involving cameras.

OpenCV cvFindHomography

In the end, it was decided to take advantage of the built-in homography matrix generation function from the OpenCV library. The inputs and output of the function had to be modified slightly to make it also compatible with two-dimensional transformations.

3.4 Entity Persistence and Averaging

Even with accurate, robust detection, there is no guarantee that every marker will be recognized in every detection frame. For example, the user's hand can briefly occlude

a marker, or some severe lighting issue like a high contrast shadow can temporarily cause erroneous detection or prevent detection altogether.

For this reason, although bit values and marker locations are calculated every frame, these are not passed directly into the application, but rather some averaging is first applied. In particular, we want to require several fames of detection before a new entity is added (in case of erroneous detection) and several frames of non-detection before an entity is removed.

To this end, a list of entities is maintained, which is updated each time detection runs on a frame. Once a specific marker has been detected in 5 frames (not necessarily consecutive) it is added as an entity. Once an entity has failed to be detected 10 times (consecutively) is is removed as an entity, and it must start the process over if it is re-added.

Additionally the position, radius, and rotation of each marker can now be averaged over a short period of time to dampen any erroneous changes.

3.5 Hardware

The criteria used to gauge the suitability of hardware can be summarized by three categories: cost, capture device, and display method.

Cost

Cost is chosen to be the quintessential limiting factor. Given unlimited funding and, thus, access to state-of-the-art hardware, the criteria in the other categories become no longer relevant.

1. **Cost** - The total cost of hardware for this system is aimed to be below $1000 CAD such that it does not restrict the user-base to professionals. Note that the budget assigned to this project assumes that the user already has a personal computer at their disposal. The cost of a personal computer is not factored into this metric.

2. **Compatibility** - Since the system is designed to use a personal computer as the main processing component, the two remaining hardware components must be compatible with most personal computers while adhering to the criteria specified in the following subsections.

Capture Device

In addition to the software components, the capture device plays a very significant role in achieving real-time results.

1. **Response Time** - Miller 1968 states that 100 milliseconds is the approximate limit for having the user feel that the system is reacting instantaneously [21]. This metric is used as the Worst-Case Execution Time (WCET) of the entire system. Ideally, the system will respond below an average of approximately 50 milliseconds. The main implication is that the capture device must be able to feed the user's actions to the detection software fast enough such that the average response time is satisfied. The maximum response time criteria for the physical capture device is set to 10 milliseconds.

2. **Capture Resolution** - One of the objectives of this system is to accurately detect a minimum of 140 markers at a given instance.The capture device must be able to capture at high enough resolution such that it allows for bit extraction to be carried out on 140 markers.

Display Method

The human visual system can detect flicker patterns at frequencies as high 120 Hz. Even at such high frame rates, judder artifacts occur from fast moving objects [22].

1. **Draw Rate** - Due to the complexity of the human visual system, it is difficult to create robust and cost effective systems which give the illusion of real-time response. Therefore, further constraints are placed on the decision metrics. In order for the system to be deemed real-time, it must be able to output the

overlaying interface at a rate of no less than 30 frames per second. 30p is the de facto standard for television broadcasting (30 frames per second without interlacing) [23]. This decision also invariably puts a restriction on the capture device. Since the overlay is processed through the GPU, rendering above 24 frames per second is not an issue as long as the capture device can supply input to the system at a faster rate.

2. **Display Method** - There are many methods of projecting an overlay onto the markers. The cost metric does away with most options involving expensive touch-sensitive table-top computers. Referring back to the 'capture resolution' metric, the overlay must be large enough such that it can project details onto the 140 markers. Furthermore, the projection method must be done in such a way that it does not interfere with the marker detection and bit extraction process.

Comparison Results

(a) Logitech QuickCam 4000 (b) Canon PowerShot (c) PlayStation Eye

Figure 3.7: Capture Devices

Common household video capture devices were the first candidates for a video capture device. The particular devices which were initially selected included a Logitech QuickCam 4000 webcam and a Canon PowerShot digital camera. The webcam produced

a lot of noise and was very sensitive to changes in ambient environmental brightness levels. It was also unable to accurately detect the required minimum number of markers (see Figure 3.8).

This device is rejected for failing the second criterion. The seven megapixel digital camera certainly satisfied this criterion, however, the software used to remotely capture images from the camera proved to be very slow. It takes on an average of 1.3 seconds for the software to trigger the camera's image and prepare it for detection. This is in conflict with the first criterion. The use of a digital camera is thus rejected.

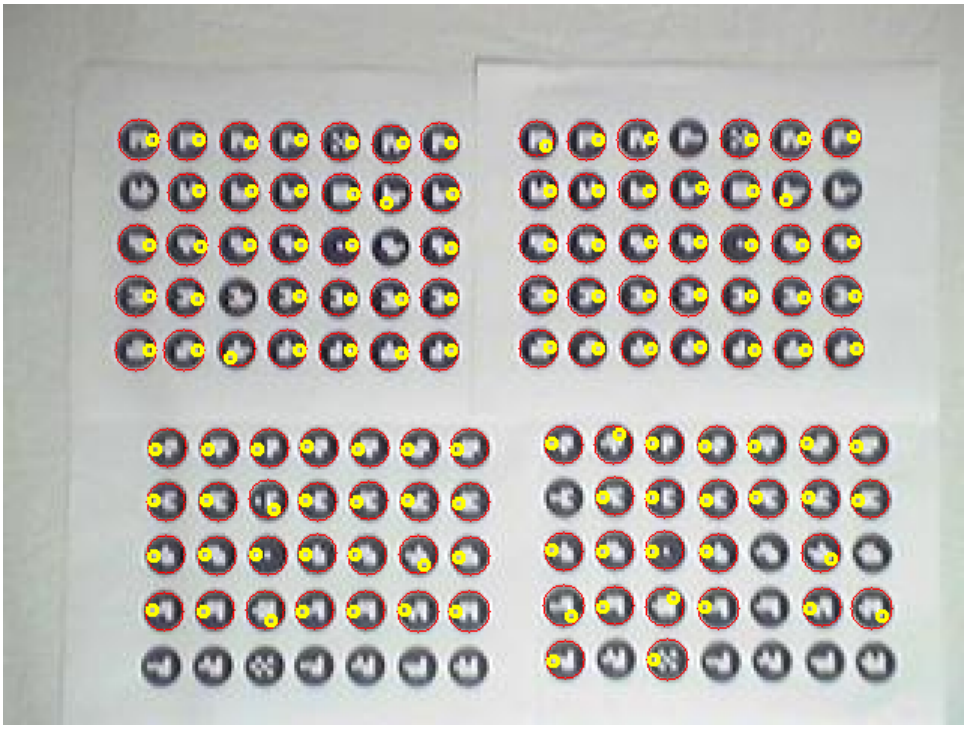


Figure 3.8: Logitech QuickCam Failing at Detection

Ultimately, the Sony PlayStation Eye camera proved to be the ideal capture device with respect to the established criteria. The PlayStation Eye is capable of capturing video of 640 by 480 pixels at a maximum rate of 75 frames per second (see Figure 3.9). The response time for the camera is well below the requirement as well. There open-source drivers available for the camera which make this device ideal for the purpose of development. With a price tag of under $45, this device is recommended over other alternatives.

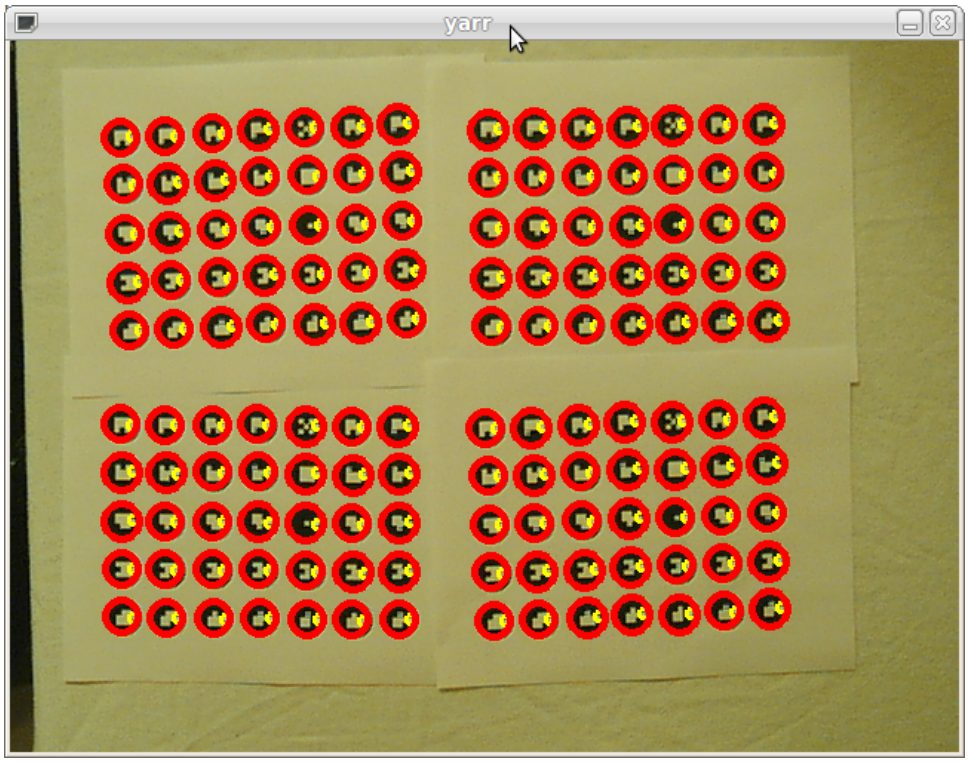
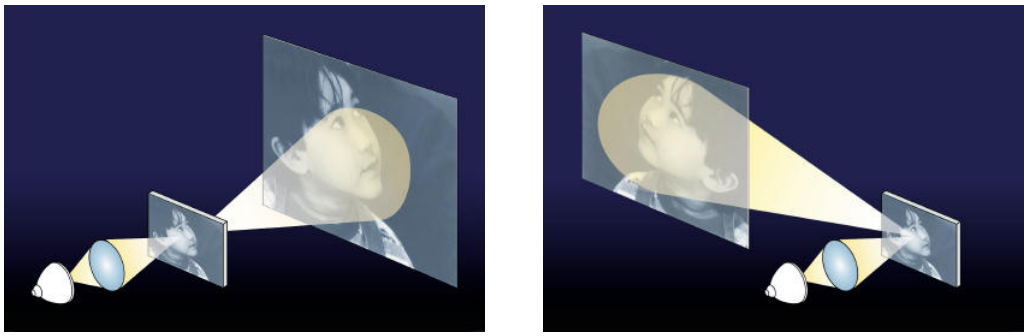


Figure 3.9: PlayStation Eye Detecting 140 markers

The idea of using a large LCD screen as a method of displaying the overlays was the first to be rejected. Although it meets all of the specified requirements, large screen LCDs are expensive. Also, the light emanating from the display onto the capture device results in glare on the camera lens which complicates the process of detection.

This leaves the option of using a projector. There are two methods of projection. Back projection (Figure 3.10b) is difficult to implement and is more expensive because it requires a custom built table and a finely calibrated mirror. Back projection also suffers from the same drawbacks of using and LCD. The other more common method is using forward projection (Figure 3.10a). By projecting on top of the surface, as well as the markers, the details of the marker would be illuminated as well, providing the high contrast required for detection and bit extraction.



(a) Forward Projection (b) Back Projection

Figure 3.10: Projector Setups

Forward projection projects high intensity light normal to the surface. Which reduces the shadows occurring from ambient lighting conditions. In conclusion, forward projection using a projector capable of displaying at a rate of 30 frames per second is ideal.

Further Conclusions

From these metrics it is deduced that the detection rate (response time) is allowed to run at a slower clock compared to the rate at which the overlay interface is redrawn. Since the overlay is processed through the GPU, rendering above 30 frames per second is not an issue as long as the capture device can supply input to the system at a faster rate. The main bottleneck occurs at the computationally expensive component of the system: marker detection. In the case that the detection rate is worse than 30 frames per second, additional corrective measures may be built into the software. Such added features include multi-threading the detection algorithm or taking advantage of a fast predictive motion estimation algorithm to adjust the overlay in between detection cycles [24].

3.6 Prototype

Figure 3.11 shows the final physical layout of the prototype. The projection device, a bright BENQ DLP projector, is mounted at the top of the apparatus. The distance at which the projector is mounted away from the table was set such that the projection area would cover at least three square feet at the lowest zoom. The mounting location of the PS3Eye camera was chosen in a similar way. Ideally, the camera would be as far away from the table as possible while covering only the projection area. This optimizes the resolution of the markers captured by the camera; it is very important to do so because high resolution increases the effectiveness of the detection and bit extraction algorithms.

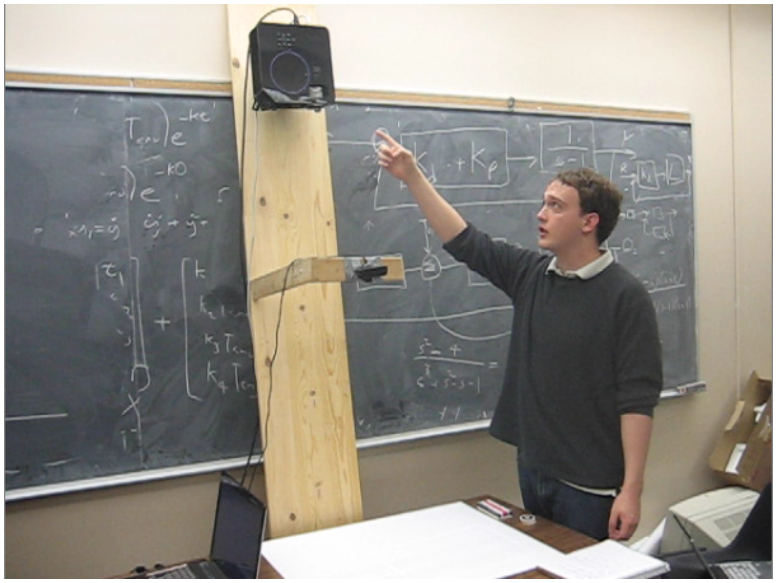


Figure 3.11: Physical Layout of the Prototype

The computer on which the algorithm is hosted is shown on the bottom left corner of the figure. The projector and camera are both connected to this computer, thus, completing the loop between the user input and visual feedback.

4

Results and Discussion

The final system and prototype design are a success. There are areas where the design could be improved and extended, but overall the design and prototype meet the goals of the project.

4.1 Objectives

- **Design a Marker:** The group successfully designed a marker which is comfortable to manipulate, and allows for robust and efficient computer recognition. The choice of a bit pattern type of recognition allowed for 255 simultaneous markers, and made the markers themselves generalizable across many applications. However, it sacrifices specificity to a single application (the pictures on the marker are not meaningful in a human sense to any particular application.)

- **Implement a Recognition Algorithm:** A marker recognition algorithm has been successfully implemented. The speed of the algorithm is very acceptable, however there is some room for improvement in the rotation and bit extraction phases. The net result after temporal averaging is a successful tactile input system.

- **Design a Visual Overlay Model:** The group successfully designed a simple application layer which successfully demonstrates the input system - a small 2D

orbital simulation sandbox.

- **Integrate Into a Physical System:** A prototype was successfully built, using a DLP projector, a PS3 eye, and a normal table.

4.2 End to End

The most meaningful high level result are the results of end to end testing. The results of these tests have been very positive; in ad-hoc testing in the lab, the orbit simulator can be readily controlled by using the spatial markers and the overall effect is as desired.. This can bee seen in the project video [25].

5

Conclusions and Recommendations

The prototype was completed on time, and as can be seen above, the primary design objectives were met. It is important, however, not to get distracted by the details, lest the overall impact of the project be lost. Thus, it is necessary to draw some conclusions as to how well the project addresses the initial problem that was identified.

5.1 Conclusions

It may be fruitful, at this point, to go back to the problem statement and evaluate the success of the design at a high level. For clarity, the problem statement is:

> Current visual design systems impose restrictions, whether physical or virtual, which create further separation between the designer and the design. Other systems, which attempt to bridge the gap between physical and virtual design, are cumbersome and restrictive. What is needed is a simple and intuitive method for integrating physical design with virtual models.

On a high level, this project succeeds in removing some restrictions which separate the user and the computer space: namely, the mouse and keyboard. The user now interacts with the computer by what would be the equivalent of reaching into the computer and moving things around herself. Furthermore, after the system is

calibrated there are not the inconveniences posed by other such tactile input systems. In this system, the user gets the virtual space projected back onto the physical space, and doesn't have to fiddle with things like wooden structures that they have to build themselves. Thus, from a very high level, the problem has been solved.

5.2 Recommendations

The problem has been solved, but caveats abound. The first is that the underlying algorithms for extracting marker information still display some instability. Especially in terms of bit extraction, the algorithms sometime mis-identify the inner bit pattern, and this error can propagate throughout the system, leading to all kinds of problems. These errors have been minimized and dampened by the idea of object persistence and temporal averaging, but minimizing the effects of the problem is not the same as solving it. A higher resolution camera would allow more accurate detection, but better results might be arrived at by rethinking the bit extraction process entirely.

Even though the display application was not the purpose of the project, it could do more to do show off the full abilities of the system. This could include increasing the classes of markers (i.e. having 7 or 8, or even 100 types of markers that do different things). Also, the application could do more display the ability to represent many different shapes and sizes of virtual objects. One marker could represent a building almost the size of the input space, while another could be a tree scarcely bigger than the marker itself.

In terms of hardware, it would be useful to integrate multiple cameras into the detection. Currently, enough hands moving over the table will eventually occlude the markers and cause them to go undetected. Of course, in the extreme case this will always be true, but the amount of collaboration that the system is capable of sustaining could be greatly increased by detecting from multiple angles. One camera would cover the blind spot of another, and vice versa. Integrating the multiple cameras wouldn't be especially difficult, because the calibration programs for transforming input spaces already exists within the system.

6

Description of Timeline

As of the submission of this report, the project has successfully adhered to the timeline established in the design plan. The deadlines for the symposium and final report submission were pushed back two weeks after the anticipated date. The extra amount of time was spent developing additional features, such as a more robust calibration method.

The timeline for the video was also modified. The group started recording video on the 23rd of February in order to have part of the video finished for the technical presentation. Further captures were made during the symposium and the video is recompiled for the final submission.

The physical design of the system proved to be the simplest task thanks to the generous help from the Collaborative Systems Lab. It was agreed to take advantage of their front projection apparatus for the purpose of testing and demonstration. This eliminated the amount of time needed to purchase and mount a projector.

References

[1] P. Rochat, "Object manipulation and exploration in 2-to 5-month-old infants," *Developmental Psychology*, vol. 25, no. 6, pp. 871–884, 1989.

[2] V. Rideout, E. Vandewater, and E. Wartella, "Zero to Six: Electronic Media in the Lives of Infants, Toddlers and Preschoolers.," 2003. Henry J. Kaiser Family Foundation, `http://www.kff.org/entmedia/loader.cfm?url=/commonspot/security/getfile.cfm\&PageID=22754.`

[3] J. Rekimoto and Y. Ayatsuka, "Cybercode: designing augmented reality environments with visual tags," in *DARE '00: Proceedings of DARE 2000 on Designing augmented reality environments*, (New York, NY, USA), pp. 1–10, ACM, 2000.

[4] T. Lssing, R. Nielsen, A. Lykke-Olesen, and T. F. Delman, "A mixed reality game for urban planning,"

[5] H. Kato, M. Billinghurst, I. Poupyrev, K. Imamoto, and K. Tachibana, "Virtual object manipulation on a table-top ar environment," *Augmented Reality, International Symposium on*, vol. 0, p. 111, 2000.

[6] H. Ishii, E. Ben-Joseph, J. Underkoffler, L. Yeung, D. Chak, Z. Kanji, and B. Piper, "Augmented urban planning workbench: Overlaying drawings, physical models and digital simulation," in *ISMAR '02: Proceedings of the 1st International Symposium on Mixed and Augmented Reality*, (Washington, DC, USA), p. 203, IEEE Computer Society, 2002.

[7] W. Lam and S. Yuen, "Efficient technique for circle detection using hypothesis filtering and Hough transform," *IEE PROCEEDINGS VISION IMAGE AND SIGNAL PROCESSING*, vol. 143, pp. 292–300, 1996.

[8] D. Kerbyson and T. Atherton, "Circle detection using Hough transform filters," in *Image Processing and its Applications, 1995., Fifth International Conference on*, pp. 370–374, 1995.

[9] Q. Li and Y. Xie, "Randomised hough transform with error propagation for line and circle detection," *Pattern Analysis & Applications*, vol. 6, no. 1, pp. 55–64, 2003.

[10] S. Chiu and J. Liaw, "An effective voting method for circle detection," *Pattern Recognition Letters*, vol. 26, no. 2, pp. 121–133, 2005.

[11] C. Ho and L. Chen, "A fast ellipse/circle detector using geometric symmetry," *Pattern Recognition*, vol. 28, no. 1, pp. 117–124, 1995.

[12] T. D'orazio, C. Guaragnella, M. Leo, and A. Distante, "A new algorithm for ball recognition using circle Hough transform and neural classifier," *Pattern Recognition*, vol. 37, no. 3, pp. 393–408, 2004.

[13] H. Yuen, J. Princen, J. Illingworth, and J. Kittler, "Comparative study of Hough transform methods for circle finding.," *Image and Vision Computing*, vol. 8, no. 1, pp. 71–77, 1990.

[14] D. Ballard, "Generalizing the Hough transform to detect arbitrary shapes," *Pattern recognition*, vol. 13, no. 2, pp. 111–122, 1981.

[15] A. Rad, K. Faez, and N. Qaragozlou, "Fast circle detection using gradient pair vectors," in *Proc. VIIth digital image computing: techniques and applications*, pp. 10–12, Citeseer, 2003.

[16] V. Ayala-Ramirez, C. Garcia-Capulin, A. Perez-Garcia, and R. Sanchez-Yanez, "Circle detection on images using genetic algorithms," *Pattern Recognition Letters*, vol. 27, no. 6, pp. 652–657, 2006.

[17] J. Xavier, M. Pacheco, D. Castro, A. Ruano, and U. Nunes, "Fast line, arc/circle and leg detection from laser scan data in a player driver," in *IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION*, vol. 4, p. 3930, Citeseer, 2005.

[18] I. Inc, "The OpenCV Open Source Computer Vision Library." Available [Online] `http://opencvlibrary.sourceforge.net`.

[19] R. Sukthankar, R. Stockton, M. Mullin, *et al.*, "Smarter presentations: Exploiting homography in camera-projector systems," in *Proceedings of International Conference on Computer Vision*, vol. 1, pp. 247–253, Citeseer, 2001.

[20] W. Fang and T. Chang, "Calibration in touch-screen systems."

[21] R. Miller, "Response time in man-computer conversational transactions," in *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*, pp. 267–277, ACM, 1968.

[22] J. Larimer and J. Gille, "5.1: Visual Performance Depends Upon Signal Resolution: Frame Rate, Dot Pitch & Bit Depth Guidelines,"

[23] D. G. Fink, "The forces at work behind the ntsc standards." Available [Online] `http://www.ntsc-tv.com/ntsc-main-01.htm`.

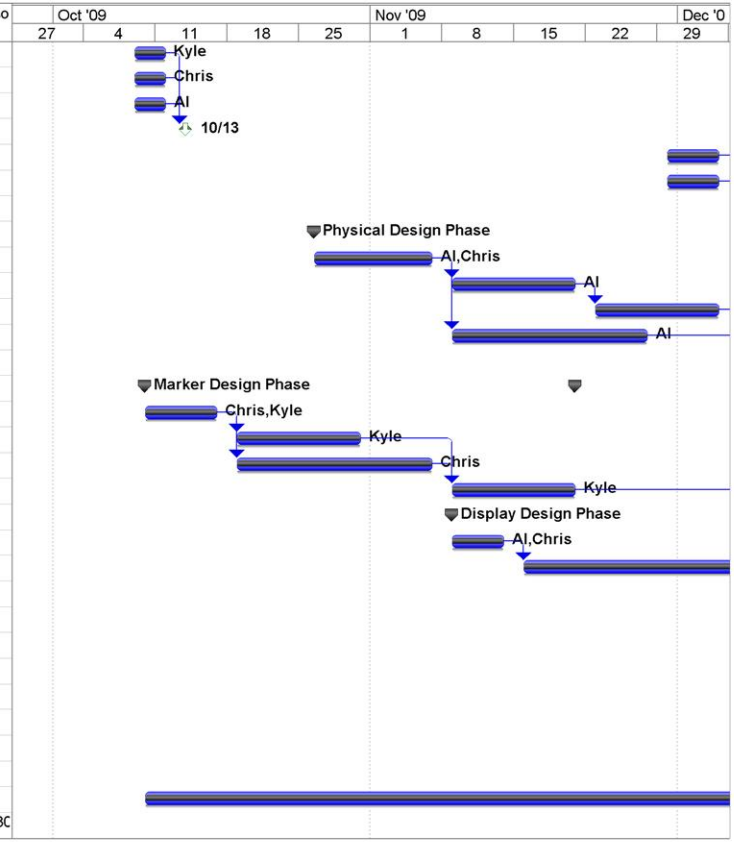
[24] D. Buzan, S. Sclaroff, and G. Kollios, "Extraction and clustering of motion trajectories in video," in *International Conference on Pattern Recognition*, pp. 521–524, 2004.

[25] A. Amir-Khalili, C. Best, and K. Morrison, "HOP Hands-On Projection (youtube video)," 2010. `http://www.youtube.com/watch?v=HigfXYnpkTM`.

Appendix A

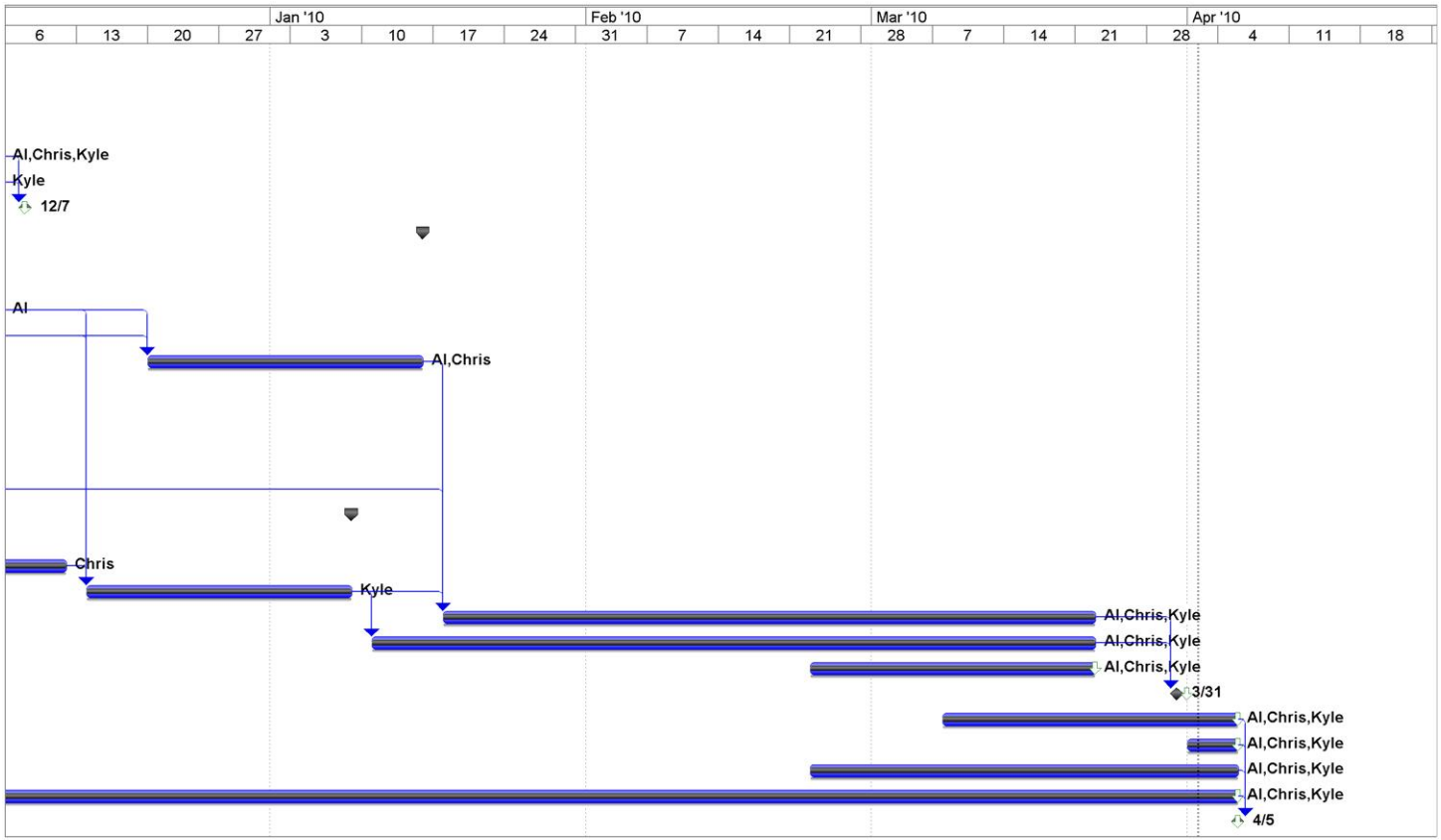
Timeline

ID	Task Name	Duration	Start	Finish	Predecesso
1	Introduction and Background	3 days?	Fri 10/9/09	Sun 10/11/09	
2	Objectives	3 days	Fri 10/9/09	Sun 10/11/09	
3	Timeline and Formatting	3 days	Fri 10/9/09	Sun 10/11/09	
4	Design Plan Submission	0 days	Tue 10/13/09	Tue 10/13/09	1,2,3
5	Progress Report	5 days?	Mon 11/30/09	Fri 12/4/09	
6	Webpage Summary	5 days?	Mon 11/30/09	Fri 12/4/09	
7	Interim Submission	0 days	Mon 12/7/09	Mon 12/7/09	5,6
8	**Physical Design Phase**	**59.5 days?**	**Mon 10/26/09**	**Fri 1/15/10**	
9	Obtain a Projector	9.5 days?	Mon 10/26/09	Fri 11/6/09	
10	Design Projector Mounting	10 days?	Mon 11/9/09	Fri 11/20/09	9
11	Build Projector Bracket	10 days	Mon 11/23/09	Fri 12/4/09	10
12	Buy/Find a Table	15 days	Mon 11/9/09	Fri 11/27/09	9
13	Mount and calibrate the pr	20 days	Sun 12/20/09	Fri 1/15/10	11,12
14	**Marker Design Phase**	**32 days?**	**Sat 10/10/09**	**Fri 11/20/09**	
15	Marker Research Phase	7 days?	Sat 10/10/09	Fri 10/16/09	
16	Marker Design	10 days?	Mon 10/19/09	Fri 10/30/09	15
17	Marker Recognition	15 days	Mon 10/19/09	Fri 11/6/09	15
18	Recognition Optimization	10 days	Mon 11/9/09	Fri 11/20/09	17,16
19	**Display Design Phase**	**45 days**	**Mon 11/9/09**	**Fri 1/8/10**	
20	Display Research Phase	5 days	Mon 11/9/09	Fri 11/13/09	
21	Display model	20 days	Mon 11/16/09	Fri 12/11/09	20
22	Overlay design	20 days	Mon 12/14/09	Fri 1/8/10	21,12,11
23	Design Finalization and Testing	46 days	Mon 1/18/10	Mon 3/22/10	18,22,13
24	Additional Feature Development	51 days	Mon 1/11/10	Mon 3/22/10	22
25	Oral Thechnical Presentation	20 days	Tue 2/23/10	Mon 3/22/10	
26	Symposium	0 days	Wed 3/31/10	Wed 3/31/10	23,24
27	Final Report	22 days	Mon 3/8/10	Mon 4/5/10	
28	Conference Summary	4 days	Thu 4/1/10	Mon 4/5/10	
29	Video	31 days	Tue 2/23/10	Mon 4/5/10	
30	Logbook	129 days?	Sat 10/10/09	Mon 4/5/10	
31	Final Submission	0 days	Mon 4/5/10	Mon 4/5/10	27,28,29,30



Project: ProjectTimeLine2
Date: Fri 4/2/10

Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline



Project: ProjectTimeLine2
Date: Fri 4/2/10

Task		Milestone		External Tasks	
Split		Summary		External Milestone	
Progress		Project Summary		Deadline	