Software Driven Multi-touch Input Display as an Improved, Intuitive, and Practical Interaction Device.

By Bradley H. Hayes
2008 Undergraduate Honors Thesis
Advised by Professor David R. Martin
Computer Science Department, Boston College

Abstract

This thesis examines and investigates the substitution of the mouse for a more natural means of human computer interaction, ranging from manipulating WIMP-based applications to developing post-WIMP interfaces and exploring their usefulness. The WIMP (Window, Icon, Menu, Pointer) interface has been the standard paradigm for the personal computing era. Their use is optimized for the keyboard and mouse input device pair, tools that have remained fundamentally stagnant with regard to innovation for decades[1]. Accomplished through the construction of a touchscreen with variable levels of contact detection, targeted demo applications not only show the effectiveness of such an input apparatus but introduce the potential for previously unexplored levels of interaction.

The use of direct-contact manipulation provides a more natural interaction than is achieved by the mouse, avoiding the need to abstract crucial concepts such as 'selecting', 'dragging', or 'resizing'. The introduction of vision driven touch-sensitivity allows for the use of physical objects to denote distinct meanings, providing a means to create associations between physical and digital actions. Building upon this concept, gesture support is a logical and practical capability to expect from a 'direct' input device. As such, it is analyzed and implemented as a core component of the device's software.

Common difficulties with software based touch-screens include device mobility, device reliability, and poor interface software implementation. While mobility is not mitigated within this project, reliability and interface/usability design are principally addressed. Challenges addressed during the implementation of the project primarily revolved around physical limitations and performance restrictions, as the quality of algorithm necessary is inversely proportional to the quality of equipment being used.

Table of Contents

I. Introduction

The utility of “Frustrated Total Internal Reflection” (FTIR) as related to H-CI was famously demonstrated by NYU researcher Jeff Han at the 2006 TED Conference, as he demonstrated his affordable, multi-touch capable screen. Similar processes have been used in a number of applications, ranging from biometrics to robotics. Despite great advances in hardware and software, the capability for more natural, powerful human-computer interaction exists and goes unused. It is the contention of this paper that the primary deterrent to adoption is subpar software interface implementation. With the development of proper interfaces this technology has the potential to surpass the standard keyboard/mouse combination in nearly every category of usability, from efficiency to ease of use.

Throughout the 2007-2008 academic year, I have constructed and designed a low budget, FTIR-driven touchscreen and have written the requisite software to provide a platform for software to leverage multi-touch development. It is crucial to note that the multi-touch nature of the screen is secondary in import to the base principle of “touch computing”. Already inexpensive and in use, single-point touchscreen display devices can offer a improvement in usability, their potential constrained by their limited interaction differentiation and the prevalence of interfaces designed strictly with traditional keyboard and mouse driven input in mind. A fundamental change in the design of functionality representation is required to optimize an application for a touch driven device. Fortunately, the target of this project is the area of laptop and desktop development rather thanthe mobile phone or PDA, as screen size is not considered a constraint. While it is not necessary to maintain a “one screen, one window” approach as Apple's iPhone effectively presents, simplification of options is of paramount importance.

It has already been demonstrated that multi-touch screens are well suited for extremely focused, specialty tasks. The objectives of the current paper are to prove its viability in a traditional setting, as well as exploring the improved potential through implementations of touch-centric

application interfaces. To accomplish the former, gesture recognition and basic touchscreen functionality is required. Regarding the latter, creative and straightforward derivations of existing interface concepts must be conceived to act as building blocks for a coherent user experience. Finally, this screen is vertically oriented for use much like a typical monitor on a desktop, rather than as a table with an interactive surface.

II. The Multi-touch Input Device

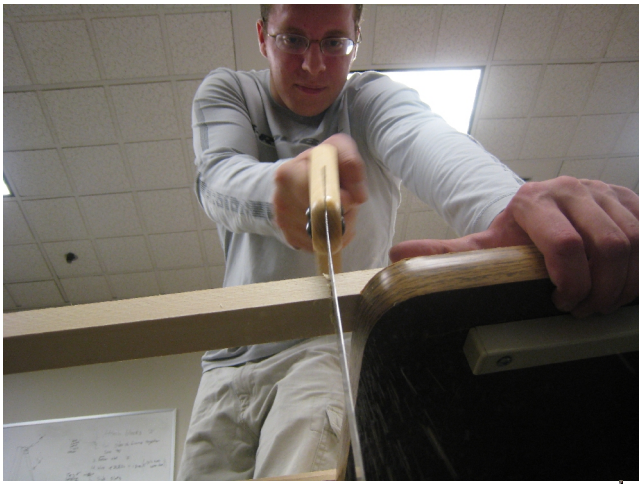
The Concept

The hardware apparatus exists to transform the task of collecting and interpreting screen contact data into a binary blob tracking problem. The actual display presentation is accomplished via rear projection, using a translucent material affixed to the screen. By utilizing rows of infrared LEDs attached to the periphery of the screen directed inwards toward the center of the glass, any object in direct contact will reflect the infrared light. This light is then processed by the webcam behind the screen, isolated by an infrared-pass filter.

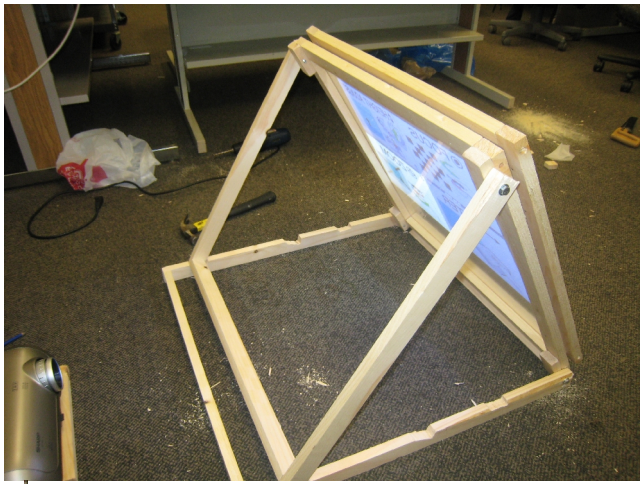
Materials

Component	Material	Notes	Approx. Cost
Screen	(2) Acrylic Panels	24”x24”x0.5” and 24”x24”x0.25”	$75
Projector	(1) Sharp XR-11XC Projector	Native 1024x768 Resolution	$450
Diffuser	(1) Sheet of Tracing Paper (1) Can of Clear, Satin Sheen Acrylic Spray	24”x24”	$10/roll $5/can
Camera	Microsoft Lifecam VX-6000		$45
IR-pass Filter	Rosco Filter Gel	Congo Blue filter color used	$5
Frame	Soft wood (Pine, Hemlock, etc.)	<u>1”x3” boards</u> (2x) Frame measures 26”x26” (1x) Base measures 32”x24” with 32” used within for additional support (1x) Stand measures 24”x24”	$35
Wiring	Standard coated copper wire	2 spools for 6 yards total (red and black colored)	$6
Light Source	High intensity IR LEDs	24 Used, 6 as spare.	$20
Adherents	Screws, Nails, Bolts, and Nuts		$15
Tools	Drill Hammer Soldering Iron Flux Multimeter		

The Construction



(a) Sawing Boards for the Frame



(b) Unreinforced Base with completed Frame

As this is a vertically oriented screen, it is necessary to design a foundation that will not shift or move under pressure from objects in contact with it. By following these instructions, it is possible to replicate the device built for this project:

1. Construct the screen frame

   First, build two identical rectangular frames to border your acrylic panel on each side. It is necessary that there be approximately an inch of clearance from the outside edge of the frame to the edge of the glass itself, as this is where the LEDs will be placed.

2. Treat the outer layer of acrylic with the polyurethane spray, allowing 15 minutes to dry. Once complete, place the diffuser between the two panes of acrylic, maintaining care to avoid creases in the material.

3. Place the acrylic and diffuser combination between each piece of the frame, using nails to fix the frames together, This has the added benefit of supporting the heavy acrylic.

   For this project, I used one nail every two inches on each side, nearly flush against the acrylic. This guarantees minimal movement of the glass within the apparatus, and makes

the frame very sturdy even if you use a weak framing material.

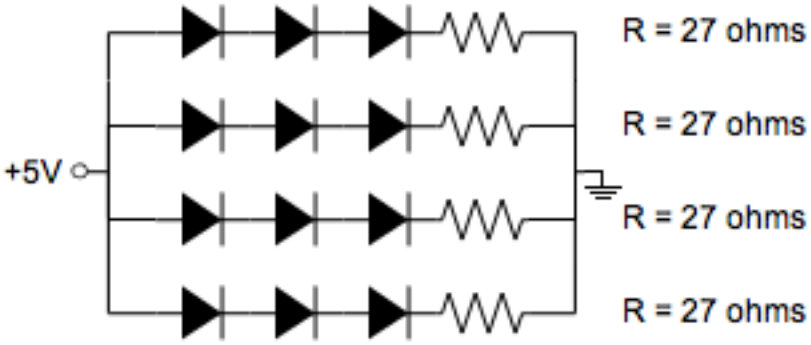
4. Build a stand to support the screen and hold the webcam. This can be as simple as creating a reinforced rectangular arrangement of wood, so long as there is support that can withstand the force applied during regular use.

   It was decided that it would be best to move forward with a variable position support, as different usage scenarios would be more conducive to different screen orientations. As such it is ideal to utilize a dual-hinged design, the support arm attached to the top of the screen frame to rest within carved notches in the base. As for the bottom of the screen frame, it is necessary to allow for rotation but not movement.

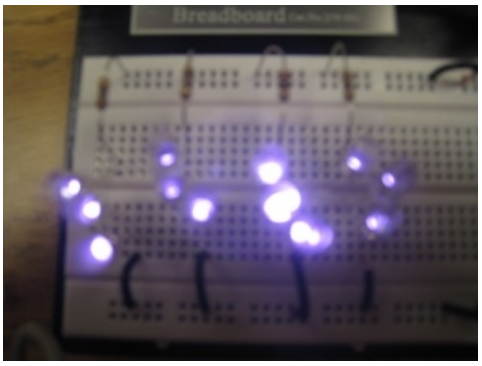
5. Carve placements in which the support arm may be seated.

   Making deep cuts into the base is important, as it would be disastrous for the screen's support to slip out from underneath it. The weight of the screen is enough to cause significant damage to any electronics that may be underneath, and the circuitry is likely to be too delicate to withstand the impact.

The Circuitry



(a) Circuit for one side of the screen



(b) Illuminated Circuit on Breadboard Through Unfiltered Camera

(c) Close-up of thin enclosure where circuits are housed

The circuits used here are very simple, but numerous. To keep the required current to a minimum, the LEDs on each side were separated into four groups of three. These groups were wired in series, connected in parallel to the power source. I recommend having at least 5 spare IR LEDs, as they are very fragile and it is easy to snap off their contacts when trying to set and manipulate them into their small housing. I strongly recommend soldering just one complete (series) circuit together and evaluating the adjustments that will be necessary to fit it into the enclosure.

Adjusting the contact leads on the LEDs is very difficult to do more than once without causing them to break, and it is almost guaranteed that you will need to cut some portion of it to neatly fit inside the screen frame. Once your measurements are made, duplicate the circuit as best as possible for the one side. Repeat this process for the other side, as it is not guaranteed that you will have the same dimensions with which to work. Additionally, the nails binding the two sides of the frame around the screen may be in the way of some of your LEDs if you are not careful with their placement.

Given the extremely confined space these circuits are being placed in, I recommend attaching the series circuits to the frames before attempting to solder the ends to power and ground. To accomplish this, the LEDs were connected via a short length of covered wire soldered to each lead, attached to the frame by staples (the wood used was soft enough to permit staples to be pushed in with pliers). In order to provide a convenient source of power and ground, I ran a cable up each side of the insides of the frames with the plastic coating stripped at key locations. The circuits were then soldered to these openings, completing the circuit with minimal frustration and difficulty. It is extremely important to remain cognizant of the location of one's fingers during this process, as it is very easy to lose focus when soldering any one of the 64 connections required. As many of these are done in very tight spaces it is easy to injure oneself with the soldering iron, so caution must be exercised.

Once the LED circuits are wired and fixed in place, power the circuit and examine it using a cell phone camera or other IR sensitive image capture device. Once it is verified that all of the LEDs are illuminated, double-check the solder joints for stability (under regular device movement/pivoting) and finally enclose the sides. It is important to be aware of the placement of the extruding power and ground lines, as a soldered connection that is broken after the device is sealed can be a hassle to repair. After inserting the bolts into the hinges, the apparatus will be fully constructed. The only issue remaining is camera placement. As can be seen in the accompanying image, the webcam is attached to the rear support of the base, orthogonally oriented with respect to the acrylic. This orthogonality is not strictly enforced, as calibration will yield a homography to remove the resulting distortions.

Preparing the Webcam

Microsoft Lifecam VX-6000

Step 1) Locate the small screw on the back of the device. Remove it and set it aside.

Step 2) Use a flat-head screwdriver or any thin, sturdy implement on hand to pry the camera open. Make sure to do this evenly around the unit, or you risk cracking the plastic. It is necessary to apply some force for the device to open.

Step 3) Find the lens apparatus and unscrew it until you can safely remove it. It is critical to avoid getting dust on the (soon to be exposed) photo sensor. If small dark dots appear on your image, it is likely caused by dust on the sensor.

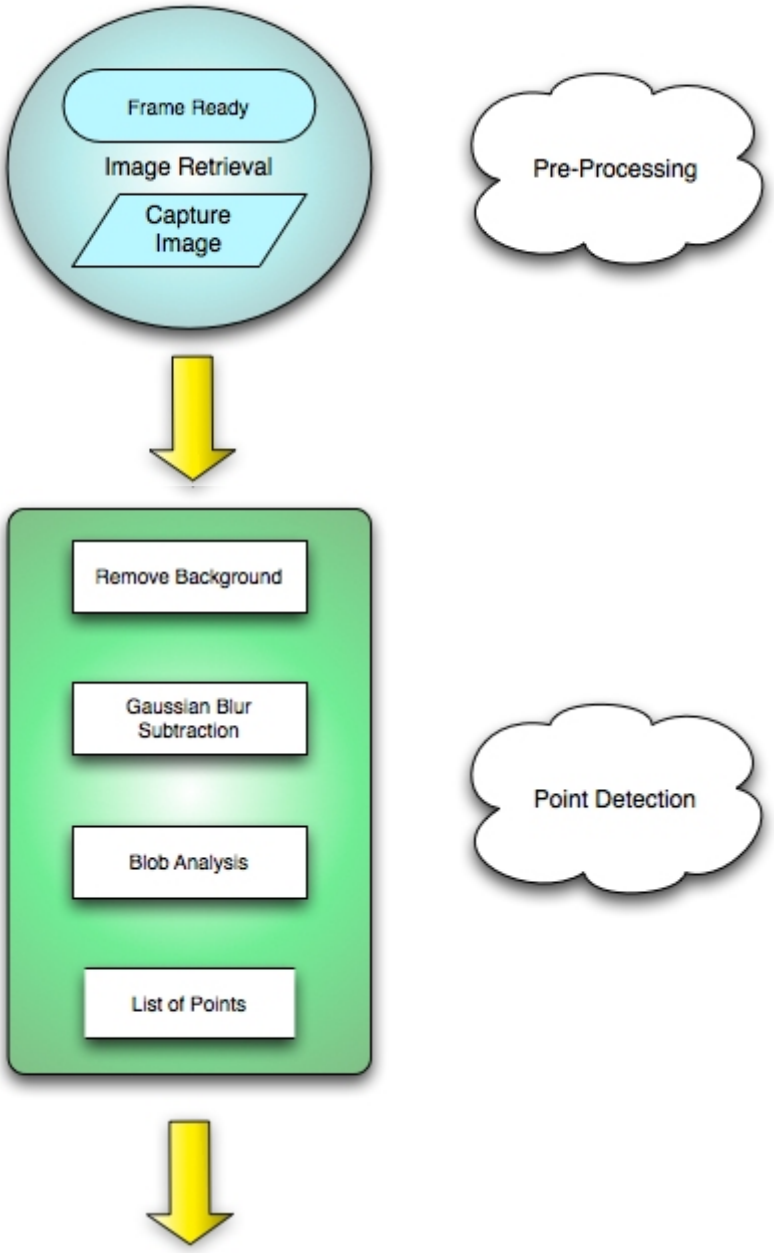
Step 4) Find the glass disc on the rear of the lens apparatus with a red hue. This is the infrared filter, and is very hard to remove intact. It is not necessary to remove in one piece as it will not be used again, but one must be wary of scratching the lens underneath.

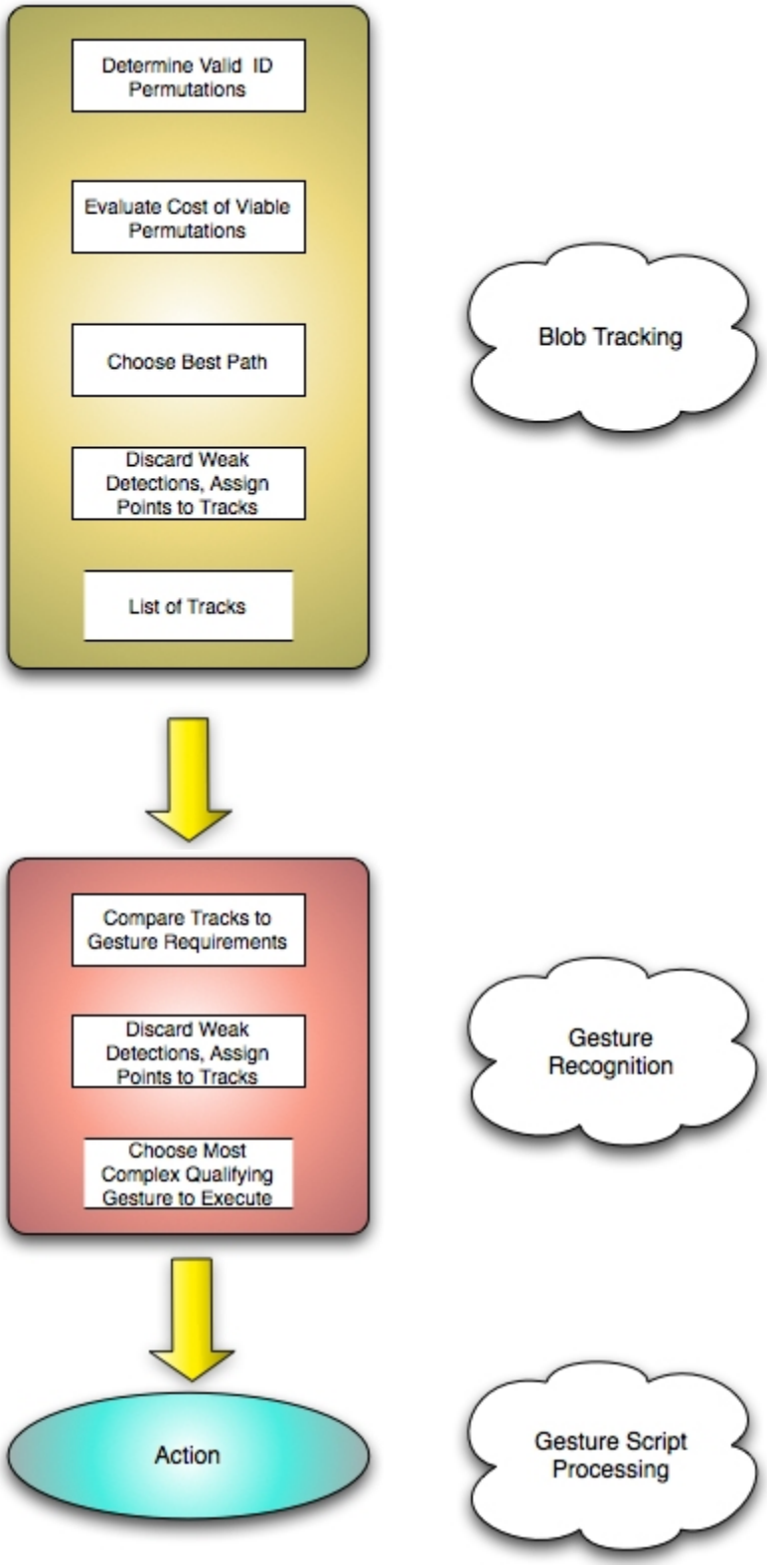
Step 5) Once the filter is removed, simply reassemble the camera.

Step 6) Using the IR-pass filter, affix a stack of 5 to 10 layers to the outer lens of the camera. This will only allow infrared light to pass through.

III. Touchscreen Software

Software Flow Diagram




Determine Valid ID Permutations
Evaluate Cost of Viable Permutations
Choose Best Path
Discard Weak Detections, Assign Points to Tracks
List of Tracks
Blob Tracking
Compare Tracks to Gesture Requirements
Discard Weak Detections, Assign Points to Tracks
Choose Most Complex Qualifying Gesture to Execute
Gesture Recognition
Action
Gesture Script Processing

Development Environment

This software was developed using Microsoft Visual C++ 2005. The touch software itself requires the OpenCV library, while the visualization software requires OpenGL and the DevIL image processing library.

Camera Handling

Capture

Frames are captured from the webcam at its native resolution. As we are dealing with filtered infrared light, a single channel image is sufficient to preserve all relevant data. As it is typical for the frame rate of the camera to be limited by the efficiency of the blob identification algorithms, significant increases in performance can be obtained by scaling the incoming image by a factor of one half or one quarter. This was not implemented in this particular paper, as precision was deemed more important than rapid sampling.

Calibration

Camera calibration is crucial to the usability of the device. This project accomplished calibration by utilizing an optional 8-point touch exercise on launch, and calculating a homography based on the identified blob locations and on-screen coordinates of the target areas. The application saves the homography to its local folder until the next calibration is run. This provides a more user friendly experience, as it quickly becomes tedious to calibrate each time the application is launched. This step utilizes its own routine for identifying blobs, as precision is of paramount importance.

Image Manipulation

(a) Background Subtracted, Noisy Image

(b) Gaussian Blur, sigma = 3

(c) Gaussian Blur of (b), sigma = 5

(d) Difference of (b) and (c)

Background Subtraction

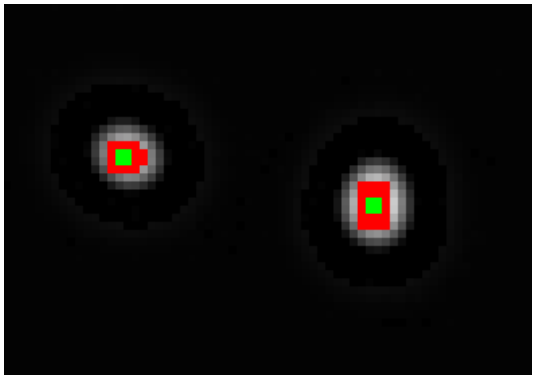
Removing the background from the image allows for an easy solution to issues arising from construction of the device. Light leakage is a common problem, as the LEDs will typically project 15 degrees from their tip in all directions. In this project, the LEDs were flush to the acrylic to maximize the incoming light for the screen itself. Incidental detection of light from the projection device is also an issue, as the projector bulb will cast intense infrared light over a small circular region which will be reflected back into the camera via the diffusion material. Background subtraction is not a complete solution to this problem, as screen tilt arises from normal usage, transforming the region vertically in the fixed camera's view. Finally, the polyurethane coating placed on the acrylic enhances the specular

property of materials in contact with the screen. This causes debris to reflect enough light to generate false positives.

Difference of Gaussian Blurs

To improve the detection of simple touch incidents, a technique must be employed to isolate peaks of intensity. By blurring the incoming camera image with a small gaussian kernel, the smoothed image gives a planar representation of the screen surface. Smooth peaks are especially helpful in that they disperse unintended reflection, an example being a part of the user's hand in close proximity to the screen surface. By blurring this smoothed image by a slightly larger gaussian kernel and consequently subtracting the two, only the peaks will remain. All other components will be eliminated by blurring the more intense peak to the weaker areas within the radius of the touch region.

Blob Detection



Analyzed Blobs, red designating most intense areas, green representing the centroid.

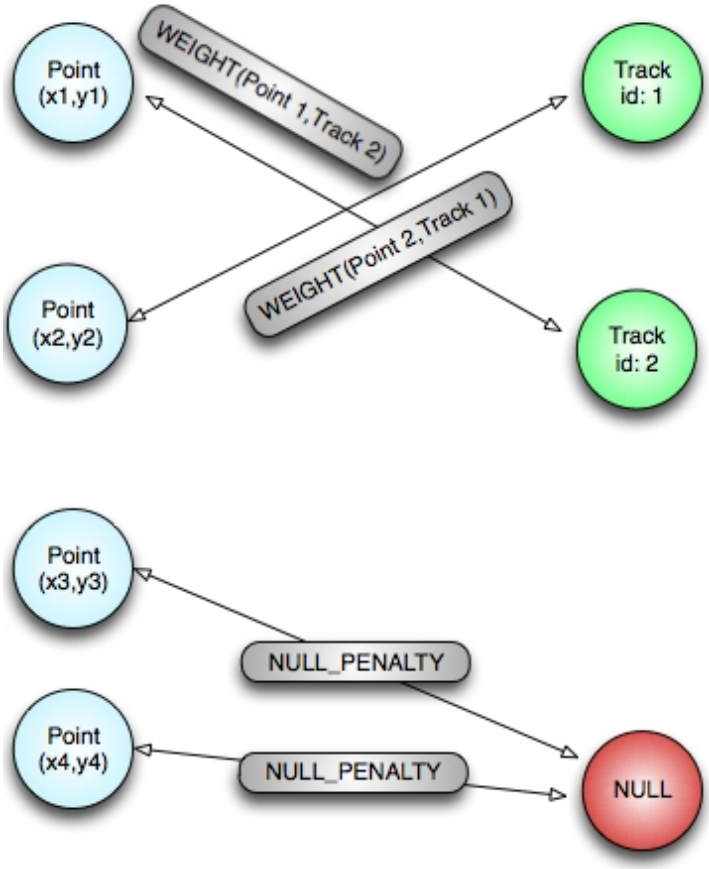
Candidate Detection

When given the processed image, the blob detection module performs a sequential pixel comparison to find edges of potential intentional contact points. Upon detection of a non-zero value, the algorithm begins a radial expansion about the coordinate. Each subsequent non-zero value has its coordinates added to a list as the expansion progresses, stopping only if a zero or value with a delta of greater than a constant ∂ = |current coordinate value – previous coordinatevalue|. As coordinates are evaluated, they are zeroed out to prevent overlapping points andinfinite recursion. Upon return from the expansion function, the centroid is calculated along with the mean intensity. Blobs found with radii outside the specified acceptable range are discarded.

Epsilon-neighborhood Point Isolation

Once a list of all candidate points are computed and filtered, they are further pared down through a process of small neighborhood elimination. If two centroids are found to be within a predetermined constant distance, that with a weaker mean intensity is discarded. This neighborhood is optimally set to be a small region. The value used for this constant is primarily affected by the delta used in the expansion function.

Track Detection



(a) Example Bipartite Graph for Point/Track Matching

Point Scoring Methodology

The most important component of the track detection algorithm is the point scoring function. When provided a point and a track to analyze, the value returned signifies the inverse confidence rating regarding the point as an extension of the specified track. Invalid points, those failing to pass basic filtering tests, are given a score of -1. As such, the point with the smallest positive value for a given track is likely to be the best match. There are two such filters, measuring the point's adherence to absolute and relative bounds.

The first filter is a simple distance filter. Should a new point lie outside a circle of radius R

centered at the track's last registered point, the scoring function will reject it as an invalid combination. The second filter performs velocity comparisons, restricting acceleration and deceleration to a certain positive percentage range. This velocity measurement occurs independent of frame rate, but rather is paced via elapsed time as calculated by the difference in clock cycles at the time of sampling. This is accurate to approximately 1/1000th of a second. Considering an optimal frame rate of 30 frames per second, or 0.033 seconds per frame, this provides an acceptable margin of error with which to reliably work.

Once invalid points are discarded, a true score is given based on the probability that the next point in the current track will occur near or at the actual region. An expected position is calculated given the history of the track. The squared distance of the potential point from the expected point is then calculated, and returned as the inverse confidence rating. If a prospective point's intensity falls beneath a certain threshold, its score is doubled as it is not necessarily deliberate and thus not considered confident.

Exhaustive Combinatorial Evaluation

Perhaps the most expensive operation in the tracking module is the evaluation of potential combinations of tracks and new points. All possible combinations of points and tracks are evaluated, each sequence having a total sum 'cost'. Points that are assigned to new track positions carry a penalty score determined by the maximum distance restriction filter. Any combination involving an impossible point and track combination, one returning a score of -1 at any point, is immediately discarded. Of those sequences that pass all filters, the lowest scoring sequence is utilized for track data.

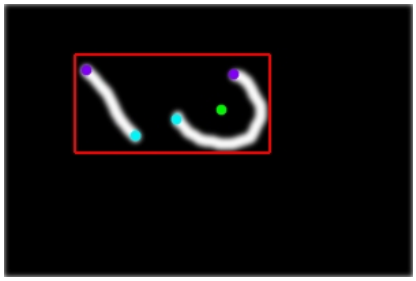
This approach transforms the tracking problem to a weighted bipartite graph. The lowest

weighted solution connecting all points and tracks (defining the creation of a new track to be a match between a point and the “null” track) is then passed on to the track creation and manipulation module. This data is utilized to make the proper connections between the tracks and points, using null-matched points as heads of new tracks. To solve the issue of unintentional contact and point registration, new points failing to have mean intensity satisfying the confidence threshold paired with the “null” track are discarded, as opposed to being identified as new tracks.

Gesture Recognition

```
Start
G SampleGesture
C 200 150 75
L 0 0 100 -1
R C:\SampleGestureProgram.exe
End
```

(a) Sample Gesture Script



(b) Gesture Analyzed Time-lapse Input (Red=Bounding Box, Purple=Track Start, Blue=Track End, Green=Center of Circular Pattern)

Gesture Expression Vocabulary

The language used to express gestures is composed of linear and circular components. The gesture author may refine these expressions by setting a variety of parameters. Linear definitions require approximate length, slope, and coordinates mapped in relation to the top-left corner of the gesture bounding box. Circular components require approximate radii in addition to the relative position data. Specific theta requirements may also be imposed, providing the ability for partial circles. While these simple definitions offer substantial variance in execution, they can easily be modified to suit significantly more demanding precision.

Action Definition

Actions are defined through the gesture document, manifesting in the form of either an executable to run, a combination of key presses to emulate, or mouse clicks to simulate. Multiple actions can be chained in succession per gesture, allowing for very deep actions. One consequence of this functionality is the ability to launch an application, manipulate menus within it, and enter data. Actions are only executed once a gesture's requirements are fully met. The addition of new classes of action is trivial.

Gesture Articulation

Gestures are defined through a simple plaintext file format. XML may make for a more user-friendly format, but its verbosity limited its utility for the project at hand. The following syntax is utilized:

Start
G <Gesture Name>
<Constraints>
<Actions>
End

Gesture Definition

The 'start' keyword signifies the beginning of a gesture definition, while the 'end' keyword closes the entry. It is typical, but not required, to place the prerequisites for execution at the start of the definition. The order of requirements is not considered during evaluation, but the order of actions dictates the order of execution.

Circular Components

The syntax for a circular track requirement is **C <X> <Y> <RADIUS IN PIXELS>**. All values must be integers, the **C** being the gesture keyword to identify the line as a circular track requirement definition.

Linear Requirement

The syntax for a linear track requirement is **L <X> <Y> <LENGTH IN PIXELS> <SLOPE>**. As with the circular syntax, all values must be integers save for the slope parameter.

Action Requirement

It is possible to run an application, simulate a mouse event, or emulate keyboard input with a gesture action.

Action Type	Identifier	Arguments	Example
Execute Application	E	(String) Path to file	E C:\myApp.exe
Mouse Emulation	M	1 – Button Down 2 – Button Up	M 1
Keyboard Emulation	K	Comma delineated list of virtual keyboard codes	K 113,122,100,32

IV. Departures From Traditional H-CI: The Adoption Problem

NYU Researcher Jefferson Han first presented his application of frustrated total internal reflection to multi-touch detection in 2005 [2]. Featured at TED, his paper and subsequent presentation vaulted multi-touch into the public eye. It can certainly be argued that the plethora of available graphical demos provide for an entertaining demonstration, yet as a consequence the common case has been woefully underdeveloped. Specialty applications are clear beneficiaries of this technology, yet the touchscreen is not readily replacing the keyboard and mouse for standarduse cases. Touch technology exists in a variety of manifestations, from “blind” touch pads and media player controls to touchscreen monitors. As the technology becomes cheaper and less delicate, it will become widely recognized as a more logical and intuitive interaction methodology. The introduction of touch-centric interfaces will provide numerous additional benefits, including cleaner interfaces, more efficient window management, and the advent of a less restrictive virtualization of physical world concepts in software applications.

Touch has necessarily failed to replace the current standard as WIMP interfaces do not translate well to touch-centric input devices. This can largely be attributed to the clutter tolerance afforded by the abstraction of the pointer from direct contact to a device represented on-screen. By utilizing clusters of small icons and thus demanding precise interactions, the interfaces typical of today's applications exhibit a steeper learning curve. Dividing these panels into separate dialogs partially alleviates the propensity for confusion, yet still introduces the same issues. By optimizing an interface for touch interaction, simplification of options and clear presentation are necessary. The utilization of touch affords many more methods of interaction than the standard click combination or drag. The object recognition potential inherently available by FTIR-based touch detection allows for physical objects to represent toolbars, dialog window activations, or other common interface component actions. This concept can be extended beyond individual application interactions, as the

blurring of the physical/digital divide allows for novel methods of desktop management as well.

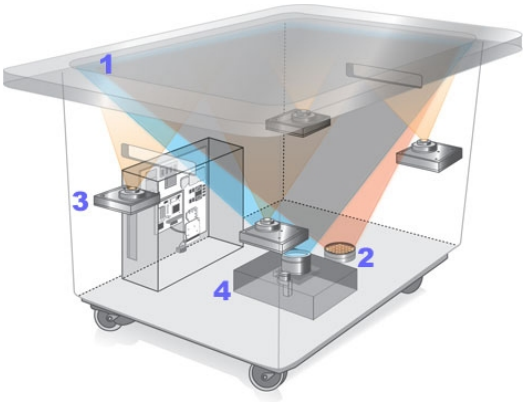
Action differentiation is vastly improved with a multi-touch capable device. Window management is a clear beneficiary of this effortless differentiation. By introducing pressure or contact point quantity as metrics, it is no longer a requirement that specific areas of windows be reserved for particular functions. The most obvious limitation of current window management is the window title bar's designation as a handle to control relocation. It is no longer necessary to create such accommodations, as an action akin to dragging three fingers from within the window's frame is quite distinct from any kind of intuitive click emulation. This logic can be extended to handle the majority of such accommodations, including the 'resize' handle, minimization and maximization buttons, and even program termination. Gesture recognition further extends the utility of multi-touch capability. Aside from a casual replacement for keyboard shortcuts, one avoids the necessary context switch between keyboard and mouse when utilizing any kind of 'sticky' action (e.g. choosing a paintbrush tool, opening a selection dialog, etc.). This substitution avoids the inevitable reduction in productivity experienced during any context switch between input devices.

Exploring the opportunities afforded by these concepts, it is evident that the abstraction of physical concepts into a digital environment is not only feasible but vital to the success of multi-touch adoption. The outdated standard of the fixed-resolution desktop has been partially reinvigorated by the concept of 'multiple desktops', yet modern implementations lack the fluidity to appeal to the common user. Given the capability to intuitively navigate over a fixed planar surface with simple hand gestures, multi-touch provides the optimal interface layer for implementing infinite resolution, continuous workspaces. With the ability to perform translations along the x-, y-, and z-axes using simple hand movements, a large-scale geography can be introduced to the desktop environment to simultaneously allow for greater collaboration and better organization.

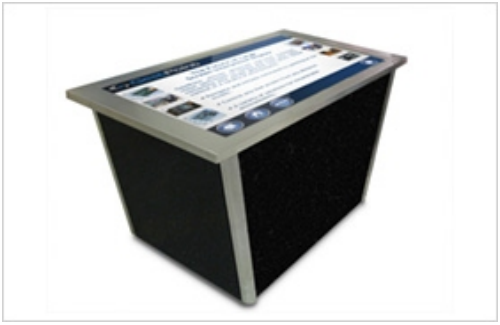
It is clear that the most substantial obstacle to multi-touch adoption in mainstream computing is that interfaces have been designed strictly for keyboard and mouse driven interaction. Taking care to design for a touch-driven device not only presents a more lucid and efficient experience, but also allows for great advancements in collaborative processes. Defining physical objects to have digital actions expands the repertoire of the interface designer in addition to dissipating the density of functionality that typically plagues complicated interfaces. While gestures may be implemented for use with the mouse, they are not as effective as those of a multi-touch screen for lack of both a unique method of distinction from normal movement and range of motions or patterns available.

V. Related Work

Multi-touch surfaces have been implemented by many, from major corporations (Microsoft, Mitsubishi, Perceptive Pixel, etc.) to hobbyist developers. As the cost of the required technology decreases, more innovation will necessarily occur in this space. William Packer of the MIT Architecture Machine Group explored this concept in the 1970's, developing a design very similar to those followed today [3]. Other inventors in this space include Mitsubishi, creators of the DiamondTouch table. This collaborative device operates differently from FTIR-based devices, accomplishing touch detection via antennas capacitively linked to users via their chairs. The DiamondTouch table is also front-projected, as opposed to most common multi-touch implementations which happen to be rear-projected [4]. Microsoft has released its own entry into the FTIR-based multi-touch field, termed the Surface. This collaborative table differs from Mitsubishi's offering in many ways, most notably the camera-driven touch detection. Utilizing five cameras underneath the screen surface, the surface offers precise position determination as well as object recognition capabilities [5]. GestureTek's Illuminate resembles the surface aesthetically and functionally, having installations throughout the world including such locations as Melbourne's Eureka Tower and New York's Time Warner building [6]. A purely entertainment-centric implementation exists in the form of the Philips Entertaible, offering advanced object tracking combined with multi-touch detection for use with the multitude of compatible games. [7] Other uses for this technology have been explored via the ReacTable, a multi-touch enabled musical interface. The surface of the ReacTable responds to various physical objects, reacting to their positioning and rotation to produce “complex and dynamic sonic topologies” [8].



Microsoft Surface [5]



GestureTek Illuminate[6]

Philips Entertaible [7]

ReacTable [8]

VI. Next Steps

Future work to be explored with this project resides in the realms of advanced gesturing, infinite resolution desktop environments, algorithm optimization, and deriving physical representations for digital actions.

Advanced Gestures

In order to avoid the need for the creation of a complicated scripting language for action definition, most complex gestures can be implemented via standalone executable files. These can then in turn be activated by a "run application" action from within the touch detection/analysis application.

Infinite Resolution Desktop Environments

The rigidly defined notion of a rendered window has steadily been redacted, with the introduction of graphically advanced window management software. One particularly relevant example is that of Beryl. Already merging 3D geographic manipulation with window management, it is a short leap to implement an infinitely large plane to contain application windows. The largest problem with such a concept is dealing with the infinitude of space, preventing lost windows. This issue can be mitigated via the introduction of bookmarks, bringing the frame to specified coordinates at a predetermined level of zoom. Additionally, the use of a geographically suggestive background may alleviate confusion. By providing users familiar representations of already-familiar geographies, an example being the solar system, it will be easier to determine window placement as well as location-function association (e.g., software development applications at location x, office productivity applications at location y, etc.)

Algorithm Optimization

As the current implementation works at frame rates in the range of 8 – 12 frames per second, the operating speed is far from the desired 30 frames per second. This is likely due to both the intense image processing required each frame and the webcam used.

Bridging the Physical/Digital Divide

The ability to perform silhouette recognition allows for previously unexamined interactions. Implementing an algorithm such as SIFT (scale-invariant feature transform) to detect and recognize objects in close proximity to the screen surface will enable physical objects to trigger actions within the touch detection/analysis software. One particular manifestation of the utility of such functionality lies in representing applications as blocks or other desktop-sized object. By placing a block against the screen for a short period of time, the touch software would analyze the object and launch the associated program, centering the application window at the location of the block. The orthogonality of the screen to the table surface limits the ability to adhere objects, carrying the advantage of preventing the user and software developers from unnecessarily cluttering the screen with physical objects.

VII. Acknowledgements

Professor Martin – You were the first Computer Science professor I had at Boston College and also one of the toughest. If it weren't for the rigor of your classes, I wouldn't have been able to complete anything nearly this complicated. The guidance provided during our weekly meetings was critical to the success of the project, especially keeping me on track after discovering that Nerf weaponry worked with the screen.

Professor Jiang – There is no question that the tracking algorithm would still be either buggy or incredibly slow without your contributions. Thank you for always making time to help me, even when your CS101 students were trying to claw through your office door.

The BC Computer Science Department – If it weren't for the great sense of community and willingness to help one another, I don't think this project would have been possible. To my peers, thank you for keeping me sane throughout this process. To my professors, your guidance throughout my four years at Boston College is invaluable and can be credited for any success that I have experienced here.

Dr. Michael Bove – Thank you for making time for me and offering guidance throughout the development of this project, pointing me towards worthwhile topics and literature.

Carl Yankowski – Your insight and experience are a great inspiration. Thank you for sharing your wisdom regarding disruptive technology and your advice on infinite resolution desktops.

VIII. References

[1] http://www-static.cc.gatech.edu/classes/cs6751_97_winter/Topics/dialog-wimp/
[2] Han, J, Low-Cost Multi-Touch Sensing through Frustrated Total Internal Reflection. In Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology, 2005
[3] http://pogue.blogs.nytimes.com/2007/03/27/the-multi-touch-screen/ William Donelson
[4] http://www.merl.com/projects/DiamondTouch/
[5] http://www.popularmechanics.com/technology/industry/4217348.html?page=2
[6] http://www.gesturetek.com/illuminate/productsolutions_illuminatetable.php
[7] http://www.research.philips.com/initiatives/entertaible/
[8] http://reactable.iua.upf.edu/