InteracTable

Authors: Suann Chi, Isha Iyer, Tanushree Mediratta

Electrical and Computer Engineering

Carnegie Mellon University

***Abstract*— This paper outlines the preliminary design of our capstone design project, InteracTable. InteracTable is a proof of concept prototype of a portable system that can turn any surface into an interactive touch screen. Current commercially available capacitive touchscreen tables are very costly and are immobile [1]. Our design overcomes these limitations by using computer vision algorithms to track the location of a user's finger and piezo sensors to detect the vibrations propagating through the medium from a tap on the table.**

***Index Terms*—Algorithms, Computer Vision, Collaboration, Detection, Interactive, Piezo Sensor, Touch Screen, Tracking**

I. Introduction

InteracTable is a portable product that will revolutionize the way people work. Any table surface can become a user's laptop screen while using the InteracTable. The size of the work surface will be adjustable to different tables, and best of all, it can accommodate multiple users for the best collaborative experience. InteracTable's design consists of a projector connected to the user's laptop, whose screen will be projected onto a flat table top. Computer vision will be used to track a red dot sticker placed on a user's finger. The projected screen will be captured by a webcam and these images are then sent back to the laptop for processing. A piezo sensor connected to a Raspberry Pi will detect a tap on the table. For this proof of concept prototype, we will constrain the system to work with only one finger.

There are existing solutions to the problem of collaboration including Google Drive and capacitive touch screen tables. Google Drive does not compare to our solution because it requires multiple computers for collaboration. Our system only requires one laptop source. Further, touch screen tables are not the greatest alternative because they cost thousands of dollars and they are not portable [1]. The InteracTable is a competitive product because it is low-cost, portable and will provide a real time response within 1 second.

II. Design Requirements

We are aiming for an accuracy rate of 95% since this is a value we have seen for many commercially available capacitive touchscreens [2]. We will verify our design by the following metrics:

1. The detected coordinate of the red dot should be within the radius of the dot. We will test this by identifying the center of the red dot for a set of test images. The distance in pixels from the center of the red dot to the detected coordinate will be calculated. Ideally, this calculated distance will be within ¾ the radius of the dot. This is a very important metric since it determines the usability of the system.
2. On detecting a coordinate, we should be able to compare the coordinate detected to the coordinates of the projected image so that we can determine whether or not a button was selected. Our GUI will have a boundary drawn around the screen so that we can use ratios to determine this metric. This testing will be done by feeding in pixel coordinates of an image of the projected GUI. These coordinates will be designated to be within the boundaries of a button or coordinates of the background screen. Our test GUI is a fixed screen so these test coordinates will not change. We will take several test images of a finger selecting a button and hovering over the background of the GUI and process them to determine whether or not the detected red dot is within the boundaries of a button on the GUI screen. A screenshot of the test GUI is provided above for reference.
3. We would like the response time for our system to be within 1 second. We will test this time by using the Python time.time() module to take the difference between the moment a tap is detected and a GUI response is triggered. This response time is very important since we want our system to be comparable to working with other collaborative tools like Google Docs.
4. The piezo sensors should detect a tap with reasonable pressure and effort. This qualitative metric can be realized with user testing. We plan to ask several users to test our system and give us feedback on how much effort it takes them to select a button. This feedback will help us tune the system so that it is reasonably responsive.

III. Architecture and/or Principle of Operation

Our system can be broken down into three main functional modules:

1. Location Detection module
2. Tap Detection module
3. Display module

The webcam would be tracking the red dot on the user's finger. When the user taps on the flat surface onto which our screen is projected, the vibrations generated by the tap will be picked up by the piezo sensors. This analog signal will then be processed by a Raspberry Pi. A basic script will be used to establish a threshold value that would indicate whether a tap occurred or not. Thus, producing a binary output, 1 being a detected tap. The binary result will then be used as a prompt for the camera to capture that frame during which this tap was detected. This frame will then be analyzed using color detection in the Lab color space, since it has the most promising results so far. The specific location will be calculated by finding the center of the red cluster that will be classified using an SVM for dynamic thresholding. After this, the detected coordinates of the red dot will be used to then calculate the corresponding coordinates in the GUI to check which button was being tapped by the user. Once this has been confirmed, a response will be generated which will lead to appropriate changes in the GUI being projected onto the surface.

We have changed our system block diagram slightly. Our latest system diagram doesn't include MATLAB anymore as we made the decision to write all of our software implementations in Python, as recommended by our course staff. This decision was made on the basis that will prevent any lag that might have occurred when transferring data between MATLAB and Python. Thus, this streamlines our entire pipeline. Fig. 1. Shows our original system block diagram which was then changed to remove MATLAB completely, as shown in Fig. 2., which shows the latest version of our system design.

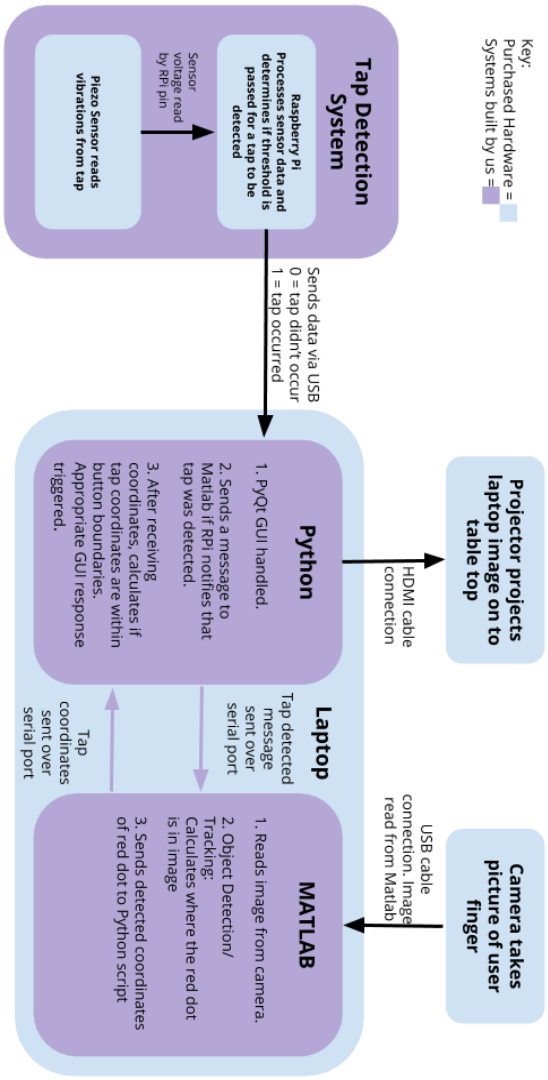

Fig. 1. Older version of our system design block diagram

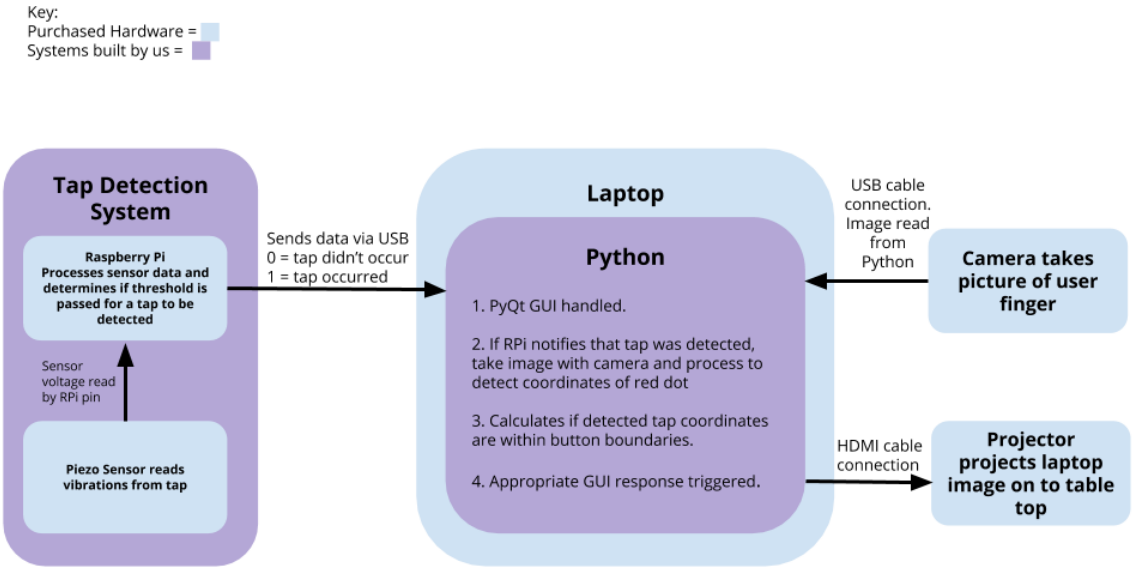

Fig. 2. Latest version of our system design block diagram

IV. Design Trade Studies

A. Tap and Location Detection Systems

i. System design selection

We had initially identified two different approaches that could potentially solve the problem of detecting where a tap occurred:

a. *Using cameras for detection and tracking algorithms*: The issue with this approach was that it required multiple cameras at different angles for us to distinguish between when the user's finger hovered over a button versus when the user actually tapped a button. We thought of overcoming this problem using Kinect to generate depth maps, however, Microsoft have stopped their production and it would only have been compatible with windows machines, which was undesirable.

b. *Using piezo sensors to detect vibrations*: This approach included finding the intersection points of several hyperbolic functions computed using waves created due to the vibrations generated by a user's tap. However, this method restricted us to a user interface that required widely spaced buttons in order for the sensors to accurately detect the location of the source of the vibration, i.e., the location of the tap.

Thus, after carefully weighing the pros and cons of the above-mentioned methods, we decided to create a system that would make use the computer vision algorithms to locate the user's finger and a piezo sensor circuit to detect whether a tap occurred or not. This way we could take advantage of both approaches while avoiding their respective drawbacks.

ii. Detection algorithm selection

Now that we had decided to use computer vision to solve the problem of localizing the user's finger, our next step was to decide on an algorithm that would allow us to accurately track the user's finger, which serves as an input to our system. The simplest solution we could think of was to place a bright red dot on his/her fingertip. Now the decision process was two-fold:

a. *Tracking versus Detection:*

We would either implement a tracking algorithm, Lucas Kanade in our case, or a detection algorithm that would find the red dot in a single frame in which the tap occurred [8]. After careful consideration, we realized that it would be computationally more efficient to localize the user's finger in one frame instead of trying to keep track of it over several frames. We found the complexity of the Lucas-Kanade Algorithm to be O(n^3) and the detection algorithms to be O(n^2) [7].

b. *Color detection versus Circle detection:*

Having concluded that we wanted to implement detection instead of tracking, our next decision required us to choose between color detection and circle detection.

Color detection seemed to be an obvious choice since the "object" that we would be detecting would be a red dot [10]. However, we did not know how the projected light rays from the projector would affect the red color of the dot. Hence, we experimented with three different color spaces- RGB, HSV, Lab. RGB completely failed as expected, whereas Lab performed the best. The preliminary results are shown in Fig. 3.

Since we were not sure how robust color detection would be, we also implemented circle detection. This algorithm was chosen because we knew that the contour of the circular dot would remain unaffected for the most part. The circle detection algorithm was a tweaked version of the Determinant of Hessian blob detection algorithm [12], [13]. The difference between our implemented algorithm and the original one was the fact that the scale of the blur was fixed since the radius of the circle was known. This reduced the iterations which would have to be made for scale selection. The preliminary results for circle detection can be seen in Fig. 4.

As a note, we are still in the process of testing with a larger set of test images. Our preliminary results include only four test images.

Based on our initial results, we have decided to stick with Lab color detection as it seems to be the most promising out of the three methods [11]. To improve the robustness of our results, we will be implementing dynamic thresholding using an SVM that would help classify red pixels versus non-red pixels. This would replace our current hard thresholding which has a high chance of failing if there is a change in the testing environment.

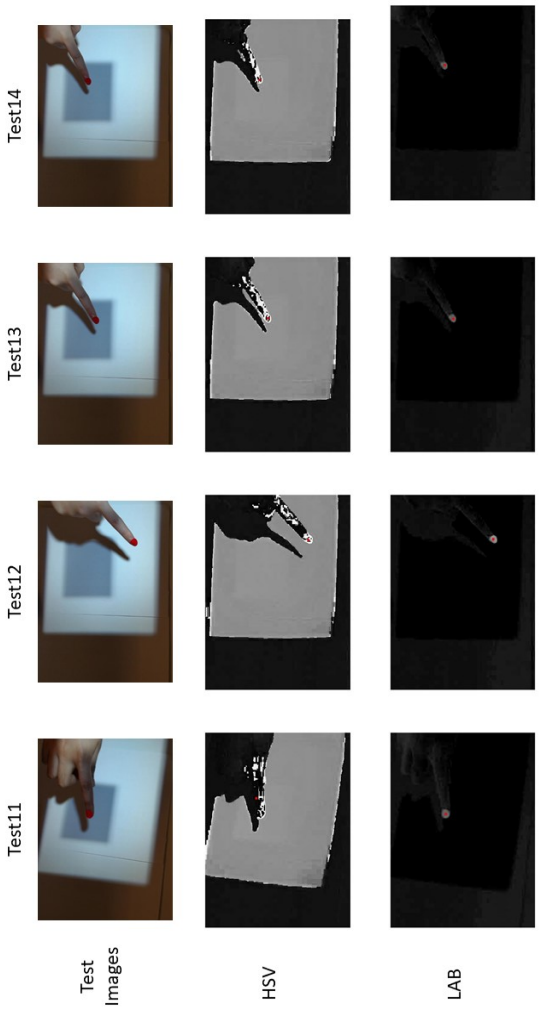

Fig. 3. Preliminary test results for HSV and Lab color spaces for color detection

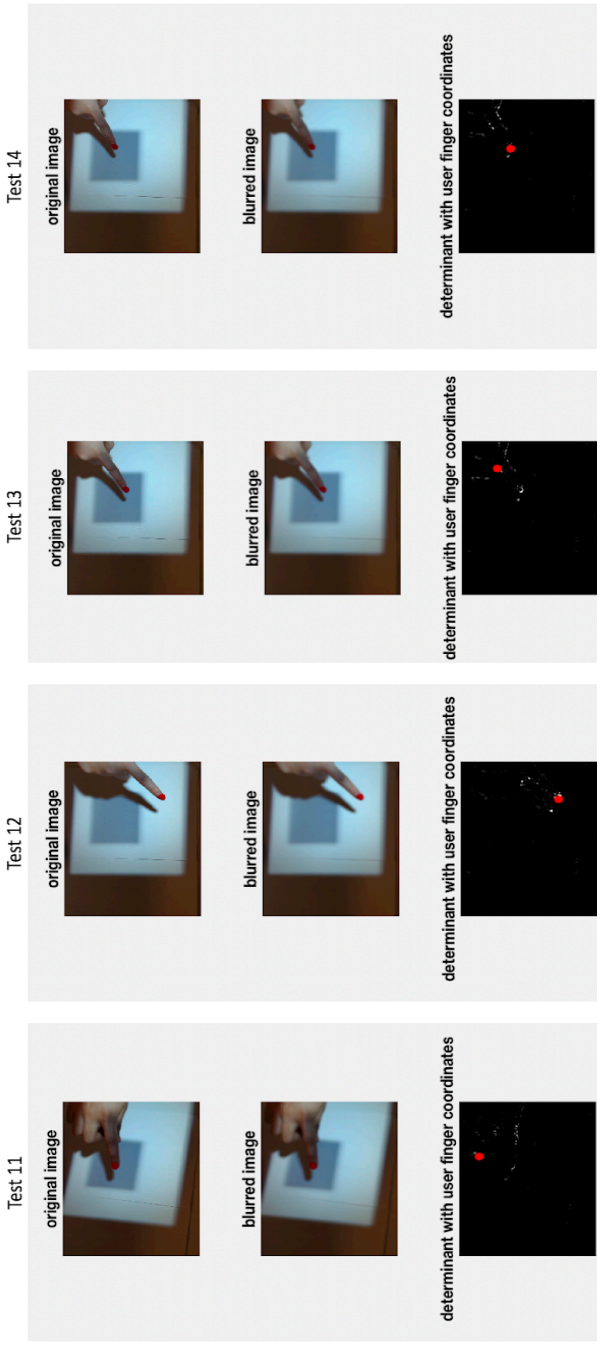

Fig. 4. Preliminary test results for circle detection

iii. Programming language selection

Our initial game plan was to implement all our computer vision algorithms in MATLAB and other data handling in Python. However, we decided to act on the feedback given to us by our staff and implement all of our code in Python. This would reduce any latency that could have surfaced when interfacing between MATLAB and Python and help us better achieve our goal of a 1 second response time.

B. Display System

GUI color scheme selection

After taking a few test images in a setup that closely mimicked our demo environment, we noticed that projecting a darker color for a button was occluding the red dot which reduced the accuracy of both our color and circle detection algorithms. The fact that we would only try to detect the location of the red dot when the user tapped on a button lead to our decision of inverting the GUI color scheme. This means that we would now have white hues for the buttons and darker hues for the background.

V. System Description

A. Tap Detection System

The tap detection system consists of the piezo sensor circuit, which will be reading user tap data and the Raspberry Pi, which will be notifying the display system when there is a user tap.

The piezo sensor circuit will need to include an ADC to deal with the analog outputs that will come from the sensor. This microcontroller is necessary because the Pi's pins cannot handle high voltage spikes that may occasionally come in from the piezo. This circuit will feed its information to the Pi's pins. The ADC microcontroller will interface with the Pi using SPI protocol [9]. SPI is adequate because this is the only device that will be connected to the Pi.

The Raspberry Pi will be installed with Python 3.7 in order to interface with the laptop's Python script. In order to receive signals from the piezo sensor circuit the Pi will continuously poll its pins. This Python script will be written using the library pigpio. To send data to the laptop, another script will be written in Python using the sockets library.

B. Location Detection System

The location detection system consists of the webcam and the laptop.

When the laptop has received tap confirmation, it triggers the webcam to take a picture of the current tabletop. This command will be sent using the OpenCV library for Python. After this Python script saves the webcam picture, it is up to a color detection program, also written in Python, to analyze the image. This script will detect whether or not the user's finger is over a button. If the answer is yes, GUI-space coordinates for the user's finger will be calculated, also using Python.

C. Display System

The display system consists of the laptop and the projector. They are connected via an HDMI to HDMI cable.

If there are GUI-space coordinates available, Python will update the GUI's state appropriately. The GUI itself will also be written in Python using the PyQt library.

The projector will display the laptop screen via the information it receives through the HDMI cable. Its focus can be adjusted depending on the distance between itself and the poster board for greater clarity. Additionally, it has a keystone to adjust for vertical tilt of the projected screen.

VI. Project Management

A. Schedule

A detailed schedule is shown in Fig. 5. on page 7. We have highlighted the four main milestones we want to achieve throughout the semester. We have already achieved milestone 1 of finalizing our design: we have chosen choosing color detection along with SVM as our detection algorithm and Python as our language to implement all software.

We have color coded the tasks indicating who is in charge of what. Yellow tasks are Isha's, pink are Suann's and green are Tanushree's tasks. Blue tasks are to be done by everyone. Further, boxes which are outline with a different color indicate secondary tasks for the respective individuals.

B. Team Member Responsibilities

Table 1. Individual task assignments

Name	Primary Completed Tasks	Primary Tasks to be Completed	Secondary Tasks
Isha	Color Detection Algorithms in Matlab. Compare Color Spaces RGB, HSV, LAB and choose best for our system Set up preliminary GUI for Matlab to Python pipeline (this pipeline is no longer a part of our design)	Color Detection implemented in Python and SVM for automating color thresholding in Python Real time detection with camera and projector setup Test for metric 1: detected coordinate is within red dot	
Tanushree	Circle Detection algorithm in Matlab.	GUI using PyQt with required functionality Reading sensor data from the piezo sensor circuit to decide threshold for tap detection Test for metric 2: detected pixel coordinate is within the GUI button	Help Suann with building the piezo sensor circuit
Suann		Piezo Sensor Circuit Raspberry pi sending data to laptop Lucas-Kanade Testing for low	Help Isha with making the detection algorithms work with our camera and projector setup.

C. Budget

Table. 2. lists all the purchases we have made so far can be found on page 8. So far, we have spent $294.52, which means that we have $305.48 left.

D. Risk Management

The following are some technical challenges and design risks we identified at the beginning of the semester.

Design Risks

We recognize that surfaces can be of varying dimensions. To accommodate these varied dimensions of our projected GUI, we will add a border at the edge of the screen that can help us calculate the coordinates of a finger tap by using the distance from the tap to the border. The border would also confine the workspace to a restricted area, thus helping us control our environment.

We were aware that we may face an issue in finding the perfect illumination under which the projected screen is clearly visible and the red dot on the finger is not obstructed by the colors of the projected screen. When buying the projector, we made sure to look for reviews of the projector being used in daylight so that we could make sure the image is clearly visible under room lights.

We were concerned about facing an issue with the camera tracking the color red since we did not know how the projector light may change the color of the red dot we are tracking. A hand may also distort the projected screen by interfering with the light rays. This is the reason why we decided to track the shape of the dot in conjunction with the color. After setting up our preliminary test environment, we quickly realized that there was negligible distortion from the hand and little to no color change in the red dot from the projected light.

We expect there may be delays in processing data in real time. We will try to resolve this delay by using multithreading and small kernels.

It may be a challenge to set up the right circuit for our piezo sensors. Before purchasing the Raspberry Pi, we were under the assumption that we could use it to read analog data just like an Arduino. It seems that that is not the case. If we run in to too many difficulties reading the analog piezo sensor input, we will switch to using an Arduino to read sensor data. In the case that we are unable to use the piezo sensors, we will fall back to using an accelerometer built in to a cheap Android phone. This will give us similar results in detecting the vibrations from a tap on our demo surface.

Risk Reduction Measures

After our design presentation, we were advised to change our design to omit the use of MATLAB alongside Python since there could be delays in sending data from Python to MATLAB and vice versa. We decided to write our detection algorithms in Python to omit any possible delays.

In order to prevent spending too much time implementing algorithms so that we can move on to other parts of the project, we set a hard deadline of March 1st to make our conclusions on which algorithm would be best to use. This deadline was to prevent the risk of falling behind schedule so that we can make sure we have a working demo by the end of the semester.

VII. Related Work

A touchscreen table is the most comparable product to our design. However, touchscreen table are very expensive - they

can cost thousands of dollars - and they are not portable [3]. Our design can be implemented at any table. If we had the budget to buy a very small projector and camera, our prototype would be easier to transport to different surfaces.

We researched other similar projects done by professors at CMU. One team of researchers made spray paint, named Electrick, that can turn any surface into a touchscreen [4]. Our design is not as permanent as paint.

Professor Chris Harrison built a device that "acoustically couples mobile devices to surfaces" using several piezo sensors [5]. The results from this research project spurred us to decide to use piezo sensors for our own implementation.

For the project OmniTouch, Professor Chris Harrison also created a handheld device that projects a screen on a person's arm and detects a tap on the buttons on the arm [6]. This localizes a tap on the table using depth maps to determine the location of a finger and if a finger taps the arm.

References

[1] "3M C4667PW 46" Projected Capacitive 60 Points Multi-Touch Display," *neweggbusiness.com*. Available: https://www.neweggbusiness.com/Product/Product.aspx?Item=9B-0WX-000M-00034&ignorebbr=1&source=region&nm_mc=KNC-GoogleBiz-PC&cm_mmc=KNC-GoogleBiz-PC-_-pla-_-Interactive+Digital+Signage-_-9B-0WX-000M-00034&gclid=EAIaIQobChMI_eDzvfDo4AIVjRyGCh2O0gvPEAkYAiABEgJOZ_D_BwE&gclsrc=aw.ds. [Accessed Mar. 4 2019]

[2] S. Hoober, "Common Misconceptions About Touch," *uxmatters.com,* Mar. 18, 2013. [Online]. Available:https://www.uxmatters.com/mt/archives/2013/03/common-misconceptions-about-touch.php. [Accessed Mar. 4 2019].

[3] "touchscreen table," *google.com*. Available: https://www.google.com/search?q=touchscreen+table&client=ubuntu&hs=gcy&channel=fs&source=lnms&tbm=shop&sa=X&ved=0ahUKEwj2863BxeLgAhXkqFkKHeIQCcYQ_AUIDigB&biw=773&bih=795. [Accessed Mar. 4 2019].

[4] A. Liszewski, "Scientists Figure Out How to Turn Anything Into a Touchscreen Using Conductive Spray Paint" *Gizmodo*, May 8, 2017. [Online], Available: https://gizmodo.com/scientists-figure-out-how-to-turn-anything-into-a-touch-1795016303. [Accessed Mar. 4, 2019].

[5] Xiao, R., Lew, G., Marsanico, J., Hariharan, D., Hudson, S., and Harrison, C. 2014. Toffee: Enabling Ad Hoc, Around-Device Interaction with Acoustic Time-of-Arrival Correlation. In Proceedings of the 16th International Conference on Human-Computer Interaction with Mobile Devices and Services (Toronto, Canada, September 23 - 26, 2014). MobileHCI '14. ACM, New York, NY. 67-76. [Online], Available: http://chrisharrison.net/projects/toffee/ToffeeCMU.pdf. [Accessed Mar. 4, 2019].

[6] Harrison, C., Benko, H., and Wilson, A. D. 2011. OmniTouch: Wearable Multitouch Interaction Everywhere. In Proceedings of the 24th Annual ACM Symposium on User interface Software and Technology (Santa Barbara, California, October 16 - 19, 2011). UIST '11. ACM, New York, NY. 441-450. [Online], Available: http://chrisharrison.net/projects/omnitouch/omnitouch.pdf. [Accessed Mar. 4, 2019].

[7] "What is the computational complexity of Lucas-Kanade algorithm?" *stackoverflow.com,* Mar. 14, 2014. [Online]. Available: https://stackoverflow.com/questions/21111318/what-is-the-computational-complexity-of-lucas-kanade-algorithm. [Accessed Mar. 3, 2019].

[8] Levin, G. "Computer Vision for Artists and Designers: Pedagogic Tools and Techniques for Novice Programmers". *Journal of Artificial Intelligence and Society*, Vol. 20.4. Springer Verlag, 2006. [Online]. Available: http://www.flong.com/texts/essays/essay_cvad/. [Accessed Mar. 3, 2019].

[9] M. Grusin, "Serial Peripheral Interface (SPI)," *learn.sparkfun.com*. Available: https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all. [Accessed Mar. 4, 2019].

[10] U. Sinha, "Color Spaces," [Online]. Available: http://www.aishack.in/tutorials/color-spaces-2/ [Accessed Mar. 4, 2019].

[11] MathWorks, "Understanding Color Spaces and Color Space Conversion," *mathworks.com.* Available: https://www.mathworks.com/help/images/understanding-color-spaces-and-color-space-conversion.html. [Accessed Mar. 4, 2019].

[12] Wikipedia, "Blob detection," *wikipedia.org*. [Online]. Available: https://en.wikipedia.org/wiki/Blob_detection#The_determinant_of_the_Hessian. [Accessed Mar. 4, 2019].

[13] Jan Sellner, "Introduction to the Hessian feature detector for finding blobs in an image," August, 2017. [Online]. Available: https://milania.de/blog/Introduction_to_the_Hessian_feature_detector_for_finding_blobs_in_an_image. [Accessed Mar. 4, 2019].

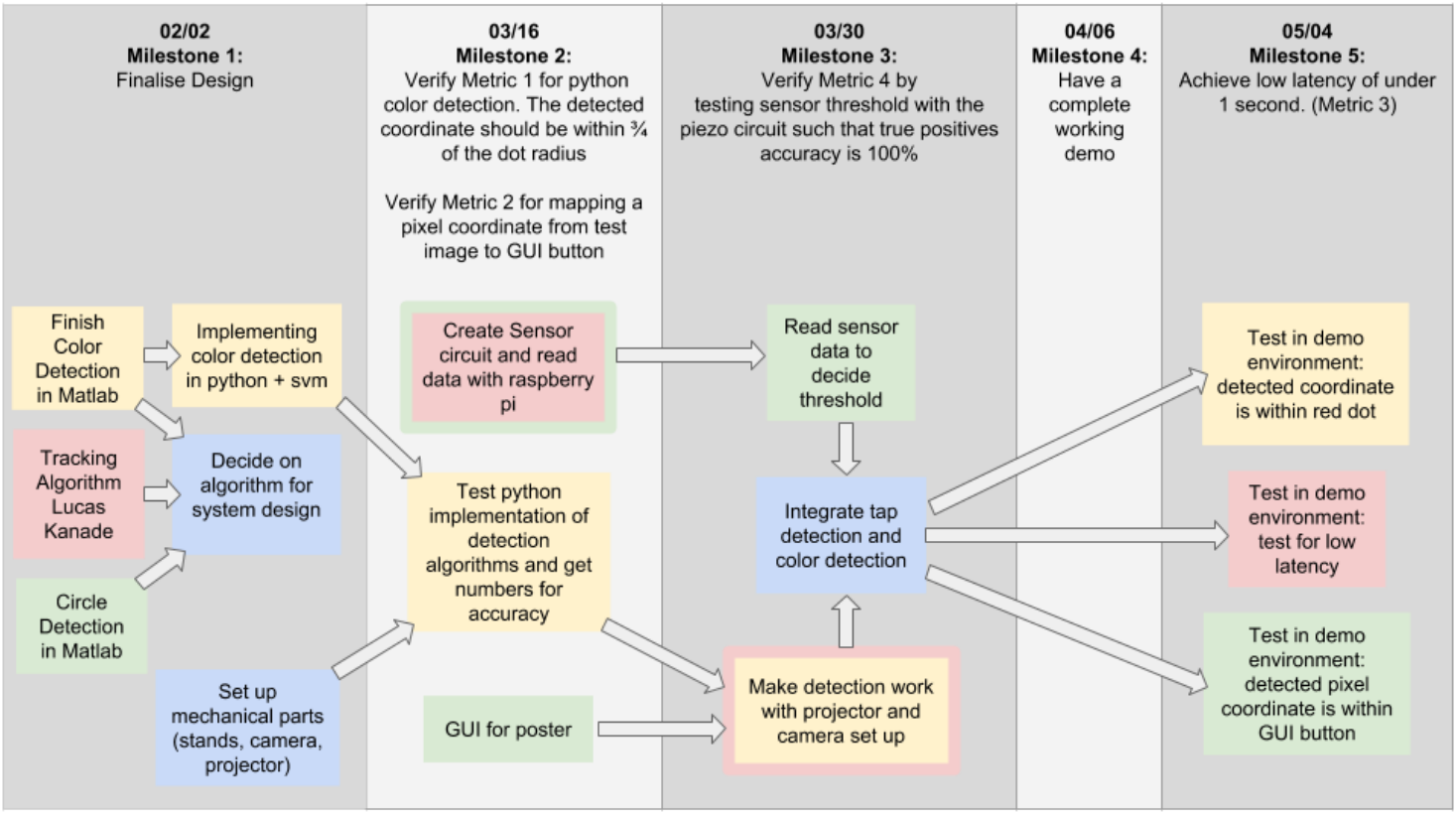

Fig. 5. Schedule along with milestones and task dependencies

Table 2. List of parts

Item	Cost	Qty	Source
Webcam	$67.93	1	Amazon
Projector (Black)	$79.99	1	Amazon
Camera Tripod (1 pack)	$6.99	1	Amazon
Projector Tripod (50 inch tripod only)	$14.99	1	Amazon
Red Dot Sticker	$7.99	1	Amazon
USB Hub (4-port)	$9.49	1	Amazon
USB Hub (Anker)	$9.99	1	Amazon
Raspberry Pi	$34.49	1	Amazon
Raspberry Pi Charging Cord	$8.47	1	Amazon
Sparkfun Piezo Sensors	$25.21	10	Sparkfun
Matlab 2018b	0		CMU Software License
Python	0		https://www.python.org
Scikit-image for Python colorspace conversions	0		http://scikit-image.org
Scikit-learn for Python SVM	0		https://scikit-learn.org
PyQt5 for GUI and PyQt5-sip needed for PyQt	0		https://www.riverbankcomputing.com/software/pyqt/download5
Personal Laptops	0		
Macbook HDMI adapter	$19.99	1	https://www.amazon.com/Adapter-uni-Thunderbolt-Compatible-MacBook/dp/B075V68NVR/ref=sr_1_6?keywords=MacBook+Pro+hdmi+adapter&qid=1551671525&s=electronics
Macbook USB adapter	$8.99	1	https://www.amazon.com/dp/B015Z7XE0A/ref=psdc_3015402011_t1_B01GGKYYT0
pigpio library	0		http://abyz.me.uk/rpi/pigpio/python.html
sockets library	0		https://docs.python.org/3/library/socket.html