Increasing Bitcoin Adoption In The Philippines

Roberto Basil E. Cruz
De La Salle University
roberto_cruz@dlsu.edu.ph

Dominic William B. Elayda
De La Salle University
dominic_elaydaiii@dlsu.edu.ph

Darren Karl A. Sapalo
De La Salle University
darren_sapalo@dlsu.edu.ph

ABSTRACT
Insert abstract here.

Keywords
Bit coin, Cryptocurrency, money

1. MOTIVATION
The Philippine Economy has been steadily increasing for the past decade. From 2001 to 2014, the Gross Domestic Product (GDP) of the Philippines increased annually by an average of 5.02%, with an all-time high of 8.90% in 2010 [1]. According to some analysts, the Philippines is set to become one of the largest economies in Southeast Asia by 2050 [2]. This trend in economic growth has been shown to have a negative relationship with unemployment. Low unemployment typically lead to higher value of goods and services and thus resulting in a healthier economy. This relationship often results in higher income for the nation and thus more spending power for consumers [3]. With potentially increasing expenditures, we believe there needs to be more options for consumers to spend their money. This means opening up more markets and payment methods for the Filipino people. One of these payment methods is the digital cryptocurrency known as Bitcoin. In this paper, we present a possible solution on increasing the adoption rate of Bitcoin in the Philippines. Section 2 discusses the idea of Bitcoin in general and how it compares to the different payment methods that currently exist in the market today. We will also be looking at the benefits and limitations of each. In section 3, we present our proposed solution on how to increase the adoption of Bitcoin in the Philippines today. We also go over an in-depth analysis of this and discuss the feasibility of the implementation of this solution in Section 4. Finally, Section 5 contains our concluding remarks and recommendations.

2. BACKGROUND
In this section, we give an overview of the idea behind cryptocurrencies. We also look into existing implementations of cryptocurrencies such as Bitcoin, Ripple and Litecoin and how they differ from each other. We will also look into existing mobile payment systems currently in place today as an alternative means of payment and how they compare to using cryptocurrencies.

2.1 Cryptocurrencies
Cryptocurrency is an alternative medium used in exchange of goods or services. It is a form of currency that is only stored, created and managed electronically, known as digital currency. Like the traditional medium of exchange such as money, the purpose of cryptocurrency is to be used to buy physical goods and services. It uses cryptography to create new units of the currency and to secure the transactions that take place, hence the name.

The idea of a cryptocurrency was conceptualized as early as 1998 by Wei Dai on the cypherpunks mailing list. He proposed a new form of money that relies on cryptography to control its creation and transaction, rather than having it managed by a centralized authority. The first implementation of cryptocurrency was created in 2009, known as Bitcoin[4].

One feature typically found in cryptocurrencies is its decentralized nature. Performing exchange is done by recording transactions in a public ledger. These transaction operate on a peer-to-peer network, with no centralized regulating body. This allows for pseudo-anonymous exchanges between parties.

While cryptocurrencies arrive in an electronic form, they are not the same as electronic money. The main difference is that electronic money still the same medium as the physical money, only represented in a digital form for ease of transferring. The creation of this currency is still regulated and done by a centralized, governing body - something not appliable in most cryptocurrencies.

2.1.1 Bitcoin
Bitcoin was launched in 2009 by Satoshi Nakomoto as âĂIJan alternative to fiat currenciesâĂİ [1]. Bitcoins are earned through a process called mining, in which computers use their processing power to solve complex mathematical calculations and to keep track of the many transactions added to

the ledger to receive new Bitcoins. The miners compete with each other to find the solution of the complex calculations. When a solution of the block of transaction is found (usually transactions every 10 minutes), the system that solved it is rewarded a block or 25 Bitcoins. This number of newly made Bitcoins issued is halved every four years, so by the year 2016, the amount of new Bitcoins made would be 12.5. This will be the case until there are a total of 21 million Bitcoins in the world.

This research paper studies different approaches in markerless finger tracking.

3. REVIEW OF RELATED TECHNOLOGY

Vision-based hand tracking systems [**?**, **?**] produced encouraging results. In [**?**], their finger tracking process followed the following steps:

1. **Image acquisition** is receiving a new image from the tracking device such as a camera.
2. **Image differencing** is producing segmenting from the raw image the region of interest which is the hand.
3. **Finger shape finding** is the recognition of the fingers from the segmented image.
4. **Hand posture classification** is the identification of the hand gesture configuration; e.g. fist, pointing gesture, pinch, etc.
5. **Application** is the module that will test the performance of the system.

In general, our finger tracking process follows a similar process: calibration, acquisition, segmentation, finger modeling, and the application. The hand posture classification step is removed because our system is limited to a fixed pointing gesture. This paper focuses on the segmentation and calibration processes.

There have been numerous hand tracking AR systems that have performed **segmentation** using image differencing [**?**, **?**], depth [**?**, **?**] or color thresholding [**?**, **?**] to retrieve the region of interest: the hand. From the segmented image, systems would then perform **finger modeling** to determine the hand's position, gesture, or the position of the fingertips. Some examples of hand modeling would be finger shape detection [**?**, **?**], 3D hand modeling [**?**, **?**] and pose estimation [**?**, **?**]. The aforementioned systems were able to track the hand accurately and in real-time, but some had to use markers to aid the tracking system [**?**, **?**, **?**] and had bulky setups [**?**, **?**].

3.1 Segmentation

Segmentation of an image is the process of separating a certain region of interest from the rest of the image. This involves removing noise, background clutter, or shadows so that a binary image is produced. A binary image is an image made up of pixels with each having a value of either zero (0) or one (1). A general way to produce a binary image is through thresholding, which involves inspecting each pixel in an image and setting its value to 1 if the pixel falls in the threshold value pair of $(low, high)$ and 0 otherwise. There are many different ways of performing image segmentation, but for the purpose of this study, we will be discussing smart image differencing, depth thresholding, and color segmentation.

3.1.1 Smart Image Differencing

Smart image differencing is an approach developed by von Hardenberg and Bérard [**?**] that was derived from image differencing. Image differencing is the process of comparing two images and detecting any difference or change between their pixels, and then generating a new image based on the detected differences of the pixels. The first image is referred to as the reference image, which in the context of finger tracking is usually the first frame captured by the camera. The reference image is then compared to the succeeding images or frames from the video stream being captured by the camera. [**?**] built upon this approach with smart image differencing. Instead of the reference image constantly being the first image captured by the camera, smart image differencing allows the reference image to learn the background from the captured images over time. This approach shows the best results only when the background is static and the object of interest in the foreground is constantly moving. A possible drawback from this approach is that it is assumed that the constantly moving object in the foreground does not rest in one position for more than 10 seconds. Another problem in this approach is that it is difficult to detect the difference when the object of interest in the foreground moves towards non-moving dark objects in the background.

3.1.2 Depth Thresholding

Depth thresholding can be performed with hardware that makes use of infrared (IR) technology. Microsoft's KINECT has an IR sensor that can compute and track the depth of an image, as well as an RGB camera that can capture images in either a 1280x960 resolution at 12 frames per second (FPS), or a 640x480 resolution at 30 FPS. Using KINECT, a depth image output can also be acquired which is composed of different color groups or blobs which represent the parts of the image that have different levels of depth. Raheja, et. al. [**?**] conducted a study that leveraged the KINECT to track fingertips and the center of the palm. Once the depth image was acquired, it was segmented after applying a calculated threshold value onto the color blobs of the regions where the hand points are located.

3.1.3 Color Segmentation

Color segmentation isolates different regions of interest in an image by differentiating them by color. Given a raw captured image, the default blue, green and red (BGR) colorspace is converted into the hue, saturation and value (HSV) colorspace, and the thresholding process is applied to the generated HSV image using a selected threshold. For this approach, there are three $(low, high)$ pairs of threshold values, which are for the hue, saturation, and value channels. The success of the segmentation relies heavily on the selected color threshold which is necessary to differentiate the skin color of the current user.

To determine the color threshold, a genetic algorithm was implemented to find an adequate HSV (hue-saturation-value)

color threshold for color segmentation. After a number of epochs, the algorithm chooses the best color threshold within the population based on a set of heuristics or fitness score.

3.2 Finger modeling

Once the appropriate binary image that shows the properly segmented region of interest is produced, the next step is to detect the fingertip. The following approaches use different modeling techniques to produce a model of a hand or a finger and identify the fingertip.

3.2.1 Von Hardenberg and Bérard's Finger Modeling Approach

Von Hardenberg and Bérard[**?**] presented an algorithm to find fingertip shapes given a binary image of a hand. Two features of a fingertip were discussed: first, the center of the fingertip is enclosed in a circle of filled (1) pixels, whose diameter is detemined by the width of the finger; and second, within a certain search square area surrounding the fingertip's inner circle, the fingertip is surrounded by a large number of unfilled (0) pixels, and is followed by a smaller number of filled pixels.

Given an (x, y) position and a region of interest, the algorithm adds the number of filled pixels around said (x, y) position within a circle of a certain diameter. The algorithm then checks for the following to determine whether or not the position is a fingertip: (1) there has to be enough filled pixels around the given position but it should not exceed a certain circle area; (2) there has to be a correct number of filled pixels vs. non-filled pixels within the search square; (3) the filled pixels outside the circle and along the search square have to be connected in a chain.

3.2.2 κ-curvature

The fingertip can be identified by finding the extended finger in a binary image, which is usually a protruding region in the contour of the hand. This can be determined by inspecting how much the contour curves, which can be done using the κ-curvature approach. Given a list of points in the contour, each point is inspected to see how much it curves in relation to the other points. An arbitrarily chosen κ dictates the distance of the relative points from the selected point.

It should be noted however that the space between fingers is recognized to have high curvature as well. κ-curvature can detect pointed areas of the hand but protruding regions do not necessarily mean fingers (e.g. knuckles or other crevices in the contour). A simple heuristic to avoid these kinds of problems would be to check for the distance of the tip of the finger from the center of the closed hand.

3.2.3 Convex Hull and Convexity Defects

Given the contour of a hand and its different outward and inward curvatures, a convex hull is a polygon which covers the outermost points of the contour. This is akin to putting a 2D rubber band around the contour. The basic algorithm for drawing the hull involves acquiring all the external points of the contour, then for every point, getting the next two points. If the set makes a convexity defect, then the second point is ignored and the remaining points are connected. A convexity defect is the area between the convexity hull and the inward-curving part of the contour which has not been covered by the convex hull. This is usually used to determine how many fingers are raised by the hand.

4. SYSTEM SETUP

As seen in figure **??**, the system uses mobile devices, a server such as a laptop or a computer, a router, and a projector. The mobile device is used to capture images and to perform finger tracking. This makes the setup portable and still capable of finger tracking. Moreover, using more of mobile devices the system can make a larger tracking space by tracking different physical spaces. Next is the server. It runs the game application and communicates with the clients (mobile devices), sending and receiving data to each other. Also the server is responsible for the visualization of the virtual space. Lastly, a router is used to connect the devices to one another. Its main function is connecting the clients to the server.

5. CONVEX HULL FINGER TRACKING

From the contour of the segmented image, a convex hull is created. Dense areas of points of the convex hull are then detected using the exterior points. With these areas, a minimum spanning ellipse can be formed around the convex hull, and the two farthest points from the center of the ellipse would be considered as a finger. After that, the tracking space is limited to a smaller set so that area near the edge of image is not considered, and the last point remaining would be the fingertip. This approach deemed effective since our setup uses only a pointing hand gesture, but can be problematic if an open hand would be tracked.

6. EVALUATION

It is important to be able to measure the performance of the different processes of finger tracking so that the system can be evaluated, providing direction for improving the system. The color segmentation and the finger tracking approach were identified to be crucial processes in finger tracking. For the scope of this paper, the evaluation methodology of the finger tracking approach will be discussed in section 6.2.

6.1 Annotated dataset

To be able to automate the performance evaluation of our finger tracking system, a dataset containing one hundred fifty (150) images of hands with the pointing gesture of ten (10) participants were collected and manually annotated. The participants were mostly Filipino and Chinese college students who had skin color ranging from dark brown to yellow. The annotation is stored in the file name, which contains the following format: **ID X x Y y.png**. Below is an example file name.

$$1\ X\ 307\ Y\ 274.png$$

The *ID* is a unique identifier of the image and x, y are the coordinate points of the actual finger tip found in the image. See Appendix A for a sample image from the dataset.

6.2 Finger tracking

Throughout the study, different finger tracking approaches were studied to see which approach would be feasible on the mobile device. With a *FingerTracking* abstract class, different tracking approaches can be implemented and evaluated. To evaluate a selected finger tracking approach, a *TrackingTester* class was developed which loads the annotated dataset containing both images and the actual point. The selected finger tracking approach is then evaluated by checking to see the distance between the **actual point** and the **detected point**. The *actual point* is the annotated point on the dataset, and the *detected point* is the result of the finger tracking approach used on the image. The *distance* is the euclidean distance between the two points and *threshold* is an arbitrary value that dictates how many pixels the detected point can vary from the actual point. The score of the tracking approach on a single image is described by the equation below:

$$Score = 0 < \frac{threshold}{distance} < 1 \quad (1)$$

Because of this design, the different tracking approaches can be evaluated rapidly, as well as different configurations of a single tracking approach. An example of different configurations could be a varying threshold values or varying rate of learning for the reference image in a smart image differencing approach. This automated evaluation aids in the selection of the best configuration for a single tracking approach.

7. RESULTS

Table 1: Performance of Convex Hull Approach

	Tile size			
Threshold	3x3	4x4	5x5	6x6
1	66.16%	77.62%	84.56%	87.95%
2	89.13%	93.57%	94.83%	95.51%
3	94.7%	95.5%	95.85%	96.02%

To test the performance of our approach, numerous tests were conducted where the threshold for the scoring was given a value of 1, 2 and 3. Furthermore, the tracking space was rescaled into a smaller resolution. A pixel in the newly scaled image would the size of a 3x3, 4x4, 5x5, or 6x6 dimension of pixels of the original image. The scores of each image would then be averaged for each test.

As seen in table 1, both the tile size and threshold affect the score of the finger tracking. Most of the tests scored at least 80.0%, which is a promising result for the convex hull tracking approach. Moreover, out of the 150 images from the dataset, only six (6) had an internal tracking score less than 20.0%. It should be noted that this approach may be effective for this specific setup, since only a pointed finger is being tracked.

8. CONCLUSIONS

The finger was successfully tracked using an approach specifically addressing the shape of a pointed finger with an accuracy of at least 80%. There can be optimizations for the finger tracking algorithm that could further increase the performance of the system in terms of accuracy.

9. ACKNOWLEDGMENTS

We would like to thank our thesis adviser, Sir Solomon See, for guiding us throughout the course of our research.

10. REFERENCES

APPENDIX

A. DATASET SAMPLE IMAGE

Figure **??** shows a sample image from the annotated dataset. As you can see, the captured image contains a hand with a pointing gesture, with the hand curled into a fist except the index finger which is stretched out. Notice that the captured image has a clear, uncluttered desk with a white background. Also note that there are shadows and changes in illumination.