– Independent Work Report Fall 2018 –


# A Smartwatch and Smartphone Based Realtime CPR Aid

Author: Daniel Greenberg - dg25@princeton.edu
Adviser: Alan Kaplan - ak18@cs.princeton.edu

## 1. Introduction and Goal

### 1.1. Introduction

CPR is an invaluable life saving tool that triples a victim's chance of survival in a medical emergency - this being the reason that it is used by medical professionals, lifeguards, and good samaritans regularly in various medical emergencies [1].

If these trained individuals were always present when an emergency arises, then they would be able to respond accordingly, and many more preventable deaths would be avoided. However, the unfortunate truth is that in a large majority of emergencies, these trained and willing individuals do not appear to be present, with a study by the National Institute of Health determining that in only about 11% of emergencies do EMS professionals record bystander help when they arrive on scene, and the AHA noting that in only 32% of cardiac arrest victims have CPR administered to them by a bystander [1, 3].

One possible reason for this lack of bystander assistance is that medical emergencies happen to occur in places where bystanders are less likely to be present. However, this does not prove to be correct; Progressive Insurance states that 77% of car accidents occur within 15 miles of one's home, and the AHA has determined that 88% of cardiac arrests happen in one's home [1, 4]. Given that the most recent US Census determined that 80.7% of Americans live in urban areas, one can deduce that these emergencies that occur in and around the home are likely to have bystanders nearby [7, 14]. Therefore, one notes that the low bystander intervention numbers are most likely not due to

a lack of bystanders being present.

Investigating this further, the real problem becomes apparent; it is not due to a lack of bystanders being present, but rather due to a lack of knowledge and training. According to the Cleveland Clinic, only about 50% of Americans know CPR, but the AHA notes that an even smaller number - only 30% - of Americans reported that they would feel confident in administering CPR when an emergency arises [1, 2]. Therefore, one notes that the reasons for low bystander assistance are most likely attributed to a lack of education and training, and not due to the lack of bystanders being present.

Therefore, a robust realtime CPR aid that is affordable and accessible to bystanders would prove to be extremely beneficial - offering immediate benefits in two distinct scenarios. The first use scenario for such an accessible realtime CPR aid would be as a supplementary training aid in CPR classes. Noting that these classes are generally many students to a single instructor, offering the students another means of feedback that is both precise and allows them to track their progress will greatly benefit their learning.

The second use scenario is during a medical emergency. In such cases, novices with limited training would be better equipped to offer life saving CPR compressions and breaths if they are guided through the steps of CPR and given feedback to make their actions maximally effective. Similarly, even if the CPR administrator is trained and experienced, the stress and external variables present during a medical emergency may lead to avoidable errors that, while simple, may drastically lower the effectiveness of the CPR administered to the victim - and therefore even these individuals would benefit from a such system [6].

### 1.2. Goal

The goal of this project project is to increase the effectiveness of bystander administered CPR, specifically by providing the bystander realtime feedback about the timing and depth of the chest compressions being administered, in addition to providing a means of reviewing CPR related data. If implemented correctly, this goal has immediate real world applications; bystanders will be able

given real life feedback and coaching to better administer CPR to a person in need, and the CPR that the victim receives will be of higher quality and effectiveness.

## 2. Background and Related Work

### 2.1. CPR Basics

CPR, while comprising of only a few steps, is heavily reliant on their being implemented correctly for it to be maximally effective [6]. Therefore, as the HeartWatch system attempts to assist with CPR administration, an overview of the subsequent steps is provided below:


**Figure 1: Infant Hand Placement [15]**

**Figure 2: Other Hand Placement [14]**

#### 2.1.1. Hand Placement

One's hand placement during CPR is crucial for effective CPR administration [1, 6, 14]. Due to the differing sizes of the chest, hand placement differs between young infants and all other victims, with both being explained below:

- Infants: Use two or three fingers of your dominant hand, kept straight, and held together. Place the fingers at the center of the infant's chest, right below the nipples [15].
- Others: For toddlers and above, a different hand placement must be used. One starts by locating the sternum (where the ribs meet), and placing two fingers of their non-dominant hand at the top of this location. Then, place the heel of one's dominant hand right above the two fingers, centered on the chest. Then, place one's non-dominant hand above the dominant hand, and interlace one's fingers. When compressing, ensure to keep one's elbows locked - using one's body weight to compress instead of bending and unbending one's arms [14].

#### 2.1.2. Compression Depth

If compressions are not done at the correct depth, then they can either be ineffective at circulating blood throughout the victim's body, or if they are done too deep then they may harm the victim. Similarly to hand placement, due to the difference in the sizes of their chests, infants have different compression requirements than all others, with both being explained below:

- Infants: Due to their chest cavity being smaller, when doing CPR on infants one should aim for compressions to be administered at 1.5 inches, or 3.81 cm. Ensure that at the end of each compression that the chest rebounds all of the way [15].
- Others: For toddlers and above, who have a deeper chest cavity, one should aim for compressions to be administered at about 2 inches, or 5.08 cm. Ensure that at the end of each compression that the chest rebounds all of the way [14].

#### 2.1.3. Compression Pace

For all victim types, the CPR administrator should aim to administer compressions at a pace of 100-120 compressions per minute, or about two compressions per second [14, 15].

#### 2.1.4. Compressions and Breaths

For all victim types, the CPR administrator should aim to administer two rescue breaths for every 30 seconds of chest compressions. One must ensure that the victim's chin is elevated when administering breaths, as this opens the victim's airway and is most effective [14, 15].

### 2.2. Related Work

Seeing as how CPR is a valuable life saving tool, it is not surprising that there are a lot of tools and research studies devoted to it. While the research and available tooling is wide ranging, that options of focus are those that are most closely related to HeartWatch's approach.

- **A New Chest Compression Depth Feedback Algorithm for High-Quality CPR Based on Smartphone (Y. Song, 2015)**: In this study, the research group investigated the viability of measuring CPR compression depth using the accelerometers available on a smartphone. Using various different combinations of denoising and integration techniques, in addition to trying

various different values for constants in the various denoising formulas, the research group succeeded in implementing an algorithm that proved to be quite accurate - with a mean error of 0.143cm, and a standard deviation of 0.100cm. However, while this algorithm proved successful, the team was limited by their using a smartphone; for the smartphone to be used to measure compressions, it needed to be placed on the victim's chest - impeding the administrator from using the correct CPR hand placement in addition to it being uncomfortable [12].

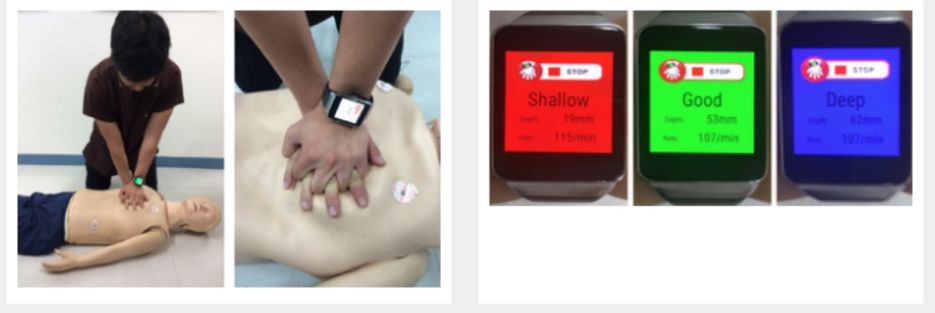


**Figure 3: Y.Song Smartwatch Approach [13]**

- **Smartwatches as Chest Compression Feedback Devices: A Feasibility Study (Y. Song, 2016)**: In this study, the same team that implemented the smartphone based CPR compression algorithm improved upon their smartphone implementation by implementing the algorithm using an Android Wear smartwatch. Using a smartwatch allowed the team to improve upon the issues that arose from the smartphone implementation, due to the fact that the smartwatch is able to be worn on the wrist - therefore allowing the administrator to use the correct CPR hand placement. Using this smartwatch implementation, the team saw even better compression accuracy than the already successful smartphone implementation. However, while this smartwatch implementation offered tangible improvements over the smartphone one, there are still downsides that this study did not addressed - the three most notable being that this study offered no means for the user to save and visualize their data, that there is no accompanying smartphone application, and that it is not open source [13].
- **Feedback Method Evaluation of CPR Support Systems under Practical Situations (R. Ohmura, 2016)**: In this study, the research team focused not on compression depth and pace, but instead

on what combination of auditory and haptic feedback CPR administrators best respond to. They administered their studies in two different scenarios; smartphone only, and smartphone and smartwatch pair. Interestingly, they noted different results depending on the system; if it was solely a smartphone, then voice feedback proves to be optimal - while if it was the dual system, then beeps accompanied with haptic feedback is optimal. While these findings offer a lot of beneficial information, this study solely focuses on the feedback given to the user - offering no instruction as to how to accurately develop such a realtime CPR application, nor evaluating the accuracy of the feedback as well [11].

In addition to research studies - like those summarized above - relating to realtime CPR evaluation, related work exists in the form of consumer devices that are already available for purchase:

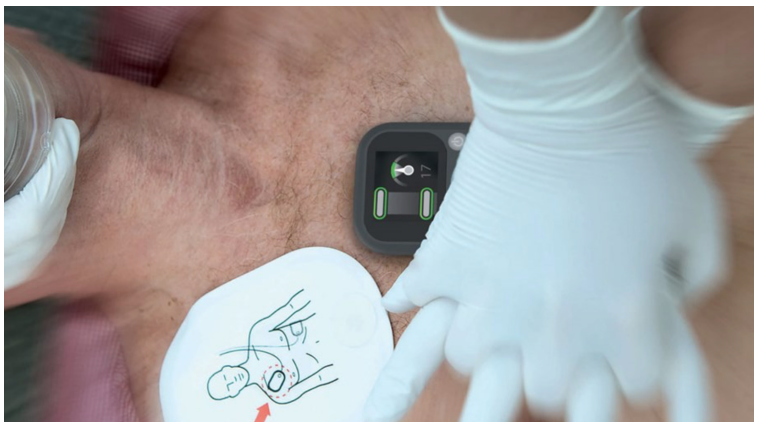

**Figure 4: CPRMeter 2 [9]**

- **Laerdal CPRMeter 2**: A moderately popular and well-reviewed option, the CPRMeter 2 is a device that uses embedded pressure sensors to communicate compression depth and pace to the CPR administrator. While this device benefits from using pressure sensors instead of accelerometers like all of the smartphone and smartwatch based approaches, it is accompanied with various downsides that the HeartWatch system attempts to address. First, since the CPRMeter 2 uses pressure sensors, it must be compressed as well, meaning that it is placed in an awkward position between the victim's chest and the CPR administrators hands. Second, this device is listed as costing 695 dollars - this price being a barrier for many to acquire this device. Lastly,

unlike a smartphone or smartwatch which many people already carry around with them, this device is not something that most bystanders will have with them on-hand - meaning that in most situations it will not be able to be used when it is needed most [9].

## 3. Approach

The Related Work section examined various research studies and products already on the market that aim, through various means, to better assist administrators in better performing CPR. This section therefore describes how the approach of the HeartWatch system builds upon the the success and addresses the shortcomings of the approaches described in the Related Work:

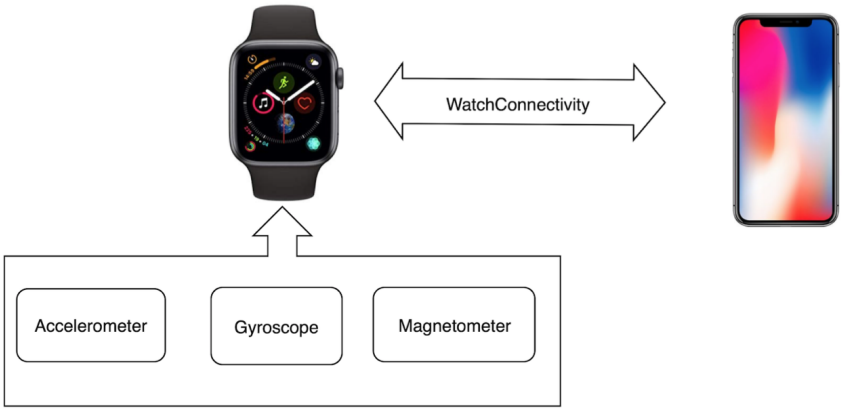


**Figure 5: Apple Watch and iPhone X**

### 3.1. Hardware

The hardware of the HeartWatch system consists of an Apple Watch Series 4 paired with an iPhone X (though any generation iPhone will work). Choosing the Apple Watch Series 4 is beneficial as it has sensors - specifically an accelerometer, gyroscope, and magnetometer - of high accuracy that allow for the system to build upon the accelerometer-based algorithms described in the related work. A smartwatch in particular also allowed for the elimination of the middle-of-the-chest placement of a smartphone-only or pressure sensor approach - allowing the user to administer CPR in a more natural fashion. Additionally, choosing a smartwatch and smartphone combination allows for the goal of greater accessibility to be met; while pressure sensor based devices like the CPRMeter 2

may be more accurate, as described above they are expensive, and the average bystander is not likely to have them in their possession when an emergency occurs [9]. Conversely, smartphones and smartwatches are growing ever more popular, and those that own them are likely to have them in their possession in many emergency situations. Lastly, the Apple Watch in particular is a wearable with wide ranging adoption - meaning that various libraries crucial for the implementation of the HeartWatch system exist with considerable community support, making the job of development considerably more manageable.

### 3.2. Software

The software for the HeartWatch system is comprised of both a full fledged watchOS and iOS applications. The watchOS application builds upon the successes of the smartwatch based approach discussed in the related work - using similar minimalist screens, audio, and haptic feedback to communicate to the user the various CPR related metrics. Where HeartWatch's software approach differs from the other approaches is predominantly through the role of the iOS application. While the other smartphone based approach necessitated the use of the smartphone for its sensors, the HeartWatch system uses an Apple Watch for this - therefore freeing up the iOS application to instead better visualize the data on a larger screen that is not oriented on one's wrist, which is largely out of view during the administration of CPR. Additionally, the use of iOS application allows for CPR session data to be saved for future evaluation - allowing the user to better be able to see trends in their CPR administration. Both of these roles are currently left largely unfulfilled by all implementations described in the Related Work section, and allow the HeartWatch system to be more robust overall.

## 4. Implementation

The implementation for the HeartWatch system consists of two key components: a watchOS application and a complimentary iOS application.

### 4.1. watchOS

The heart of the HeartWatch system is the watchOS application. First, an overview of the various UI screens and their associated functionality is provided.

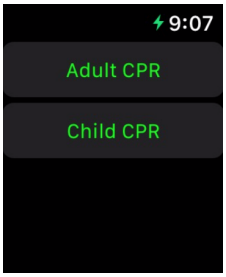


**Figure 6: Selection**

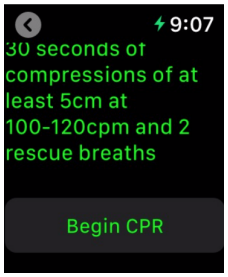


**Figure 7: Tutorial**

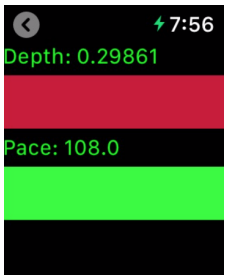


**Figure 8: Compressions**


**Figure 9: Breaths**

#### 4.1.1. Selection Screen

The first screen that the user encounters is the Selection Screen. Here, the user has the option to select the victim type for the CPR session that they are about to initiate - with the two options being "Adult" and "Child". The reason for these two differing options is that, as explained in the CPR Basics section, children and adults have differing CPR requirements, and therefore the HeartWatch system must know which type of victim CPR is being administered for. When a selection is made, the information is forwarded to the iOS application, and the user is forwarded to the next screen.

#### 4.1.2. Tutorial Screen

The second screen that is seen is the Tutorial Screen. Here, basic CPR pointers are displayed to the user, allowing them to review the necessities of CPR prior to beginning the session. When the user is ready, they click "Begin CPR" - this forwarding the information to the iOS application and forwarding the user to the next screen.

#### 4.1.3. Compressions Screen

Once CPR has been initiated, the Compressions Screen is shown to the user. When this screen is visible, the user is guided through the administration of chest compressions, with realtime feedback being provided through visual, auditory, and haptic feedback. The first helpful feedback to the user is a vibration that fires every half second - this serving to guide the user through compressions at the correct pace of 120 compressions per minute. Then, as one can see in the image, the absolute depth

and pace data is displayed to the user. Additionally, every five seconds, more in depth feedback is given to the user. If the depth value is not in the optimal range at this time, then auditory feedback is given; one beep for too shallow, and two beeps for too deep. Similarly, if the pace value is not in the optimal range, then haptic feedback is given; one vibration for too slow, and two vibrations for too fast. As mentioned, this occurs every five seconds to avoid overloading the user with feedback, since if feedback is given every half second then the user would be greatly overwhelmed. Throughout the entirety of this phase, the information is forwarded to the accompanying iOS application. This continues for 30 seconds, and then the HeartWatch system transitions to the next screen.

#### 4.1.4. Breaths Screen

After 30 seconds of compressions, the Breath Screen is shown. This screen guides the user through the administration of two rescue breaths, and then returns to the Compressions Screen for CPR to continue until the user ends the session. The Breaths Screen uses visual and auditory feedback as a means of guiding the user. When the user is meant to inhale, a blue circle is shown - being filled at the same rate that the user should be inhaling - and a single beep is heard. Then, when the user is meant to exhale, a red circle is shown - being filled at the same rate that the user should be exhaling - and two beeps are heard. During this phase, the appropriate information is forwarded to the iOS application.

Now, a technical overview of the technology behind these screens is provided. Implemented natively in Swift due to hardware constraints of the Apple Watch, the watchOS makes use of these standard technologies to function:

#### 4.1.5. CMMotionManager

*CMMotionManager* is the object for starting and managing motion services, and is part of Apple's Core Motion framework. After instantiating an instance, the HeartWatch application is able to pragmatically poll the various sensor of the Apple Watch, and, in the context of HeartWatch, supply the program with the necessary gravity-corrected acceleration data to compute CPR compression

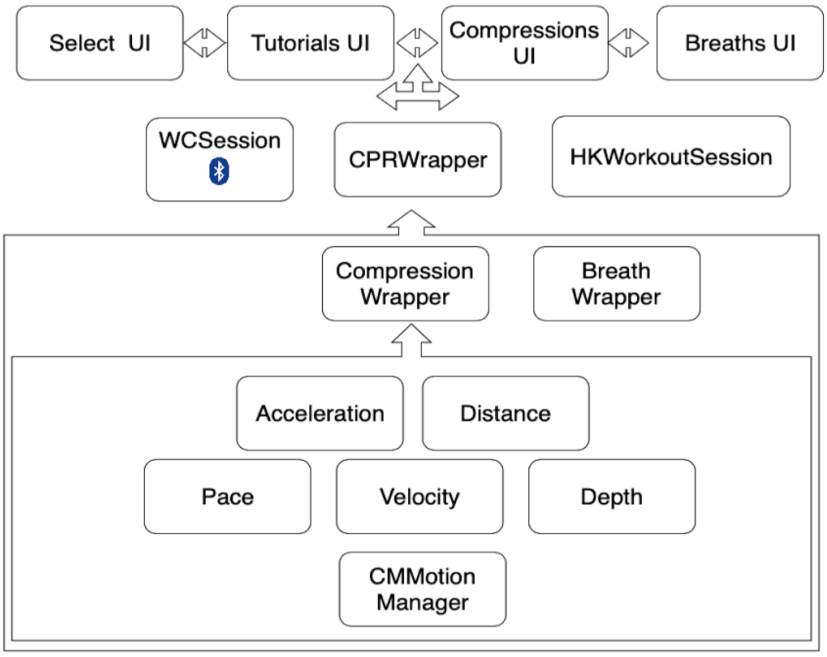


**Figure 10: watchOS Application Architecture**

depth.

### 4.1.6. WCSession

*WCSession* is the object that initiates communication between a WatchKit extension and its companion iOS app, and is part of the Watch Connectivity framework. Using an instance of this class, the watchOS application is able to determine if the Apple Watch is currently paired to an iPhone, and if the companion iOS application has been launched. If it is the case that the watchOS application is able to communicate to the iOS application, the *WCSession* object allows for messages to be sent from the Apple Watch to the iPhone, allowing the HeartWatch system to keep its various UIs in sync across the different platforms, and to send compression and pace data from the Apple Watch to the iPhone.

### 4.1.7. HKWorkoutSession

Though the HeartWatch application is not a workout application in any way, it relies heavily on the Apple Watch's HealthKit capabilities; specifically instantiating a *HKWorkoutSession* instance for

every CPR session. The reason that HeartWatch makes use of this is twofold, and both reasons are related to a watchOS application's ability to run in the background. First, after a workout is started using *HKWorkoutSession*, the Apple Watch's various sensors are ensured to keep running in the background - this ensuring that accelerometer stays awake and that the HeartWatch application is able to get the appropriate data. Second, due to hardware constraints and to prolong battery life, Apple Watch applications are not allowed to run in the background unless they start a valid workout. Therefore, to ensure that HeartWatch is able to run even when the Apple Watch's screen is off, a *HKWorkoutSession* is started to allow the application to run in the background.

#### 4.1.8. Accelerate

The *Accelerate* framework is Apple's wrapper for the popular LAPACK package; an efficient and fast linear algebra package developed in the 1990s. As described in detail later in this paper, LAPACK is crucial to the HeartWatch system as it gives the application the necessary linear algebra capabilities to efficiently and effectively denoise the acceleration data that is polled off of the device.

In addition to these standard technologies, the HeartWatch system relies on various proprietary solutions to function correctly as well. While there are many classes that the watchOS application uses, the most important are described below:

#### 4.1.9. CPRWrapper

One instance of *CPRWrapper* is created per an entire CPR session. This instance is responsible for managing all components of CPR administration - including compressions, breaths, UI changes, determining if the application is in the background, and communication with the iOS application. To accomplish all of this, the *CPRWrapper* instance does the following when the *process()* method is called:

- Keeps a state to determine if compressions or breaths should be administered, switching this state after the appropriate number of seconds.

- If the state is for compressions, the instance remakes an instance of a class name *CompressionWrapper*. *CompressionWrapper* acts to wrap all of the compression logic, including *Acceleration*, *Velocity*, etc. to allow for a uniform way of getting compression data. While processing compression data, *CPRWrapper* will update the UI if the application is in the foreground, and if the iOS app is connected it will send over the appropriate data.
- Similarly, if the state is that for breaths, the instance remakes an instance of a class named *BreathWrapper*. *BreathWrapper* simply leads the user in administering two breaths using vibrations. While processing breath data, *CPRWrapper* will update the UI if the application is in the foreground, and if the iOS app is connected it will send over the appropriate data.

#### 4.1.10. CompressionController and BreathController

These two controllers are responsible for firing the timer of a CPR session, and calling the *process()* method of the *CPRWrapper* instance every timer interval. In addition, these controllers have the various UI elements that should be updated if the watchOS app is in the foreground and visible; *CPRWrapper* takes care of checking if the application is in the foreground, and makes use of these UI elements to update the UI appropriately. It should be noted that these controllers are for accessing the UI and are not one to one with the current CPR state - that is, for instance, when the *CompressionController's* timer is fired, *CPRWrapper* may actually process breath associated data. The reason for this is that when the app is in the background, no UI state is allowed to change; therefore, if the app went into the background when *CompressionController* was visible, it cannot transition to *BreathController* without throwing an error. Therefore, the controllers blindly call *process()* in *CPRWrapper*, and that instance takes care of actually implementing CPR. Then, when the application comes into the foreground, *CPRWrapper* will update the UI if its state matches the controller, or if the state is different than the controller *CPRWrapper* will force a UI controller change.

#### 4.1.11. Acceleration

Three instances of the *Acceleration* class are created whenever compression data is being collected - one instance for each of the three primary axes. After another class in the HeartWatch system polls

the Apple Watch for acceleration data using the *CMMotionManager*, the raw values are passed to the *Acceleration* instance's *removeNoise()* method - each *Acceleration* instance getting the raw value that is appropriate for the axis that the instance is configured to manage. The *removeNoise()* method takes these raw values as input, and employs various of the LAPACK functions, provided by the Apple *Acceleration* Framework, to denoise the data - in addition to filtering the data to ensure that it does not report an acceleration that is larger than what humans can reasonably produce, as described by the L. Geddes research group [5].

#### 4.1.12. Velocity

Similarly to *Acceleration*, three instances of the *Velocity* class are created, one for each axis. These instances are passed the raw velocity values from their corresponding *Acceleration* counterparts. The main purpose of the *Velocity* class is to correct the raw velocity data, integrate, return a raw distance value, and run a heuristic to determine if a peak is detected in this axis. These various functions are discussed below:

- The technique used to correct this data is called Transient Component Emphasis, and will be discussed in detail later in the report [10]. The caller calls the *removeNoise()* method of the class with a raw velocity value for that respective axis. This method then runs Transient Component Emphasis on this raw value to get a corrected value.
- After this corrected velocity data is computed, a heuristic is computed to determine if a peak is found. To determine if a peak is found, the *checkPeak()* method is called with this new corrected value, and if a peak is found, it sets a flag to true.
- Lastly, trapezoidal integration is done to get a raw distance value from this corrected velocity value.

#### 4.1.13. Distance

Similarly to the other two classes, an instance of *Distance* is created for each axis. *Distance* does almost the same work as *Velocity*; running Transient Component Emphasis to get a corrected value. But, since this is the final value of interest, this corrected value is returned; no more work is done.

#### 4.1.14. Depth

One instance of this class is instantiated per compression session. Computing the vector sum of the raw distance values, a new distance is computed. The *Depth* instance runs another heuristic to for peak detection to go along with the ones from the *Velocity* instances. If it is determined, using all of this information, that a peak truly is found, the *Depth* instance communicates the net depth, in centimeters, and returns this value. Additionally, depending on the victim type of CPR being administered (adult or child), the *Depth* instance also communicates which color the "Depth" section of the HeartWatch UI should display.

#### 4.1.15. Pace

Similarly to *Depth*, one instance of this class is instantiated per compression session. Once *Depth* is used to determine if a peak is definitively found, *Pace* is used to determine the current pace of compressions. Via the *getPace()* method, when passed an integer representing the number of seconds that passed from the start of the compressions session, the *Pace* instance will determine a new pace value by comparing this value to the value stored from the last peak. This pace value is returned, in addition to which color the "Pace" section of the HeartWatch UI should display.

### 4.2. iOS

The companion application in the HeartWatch system, the iOS application builds upon the watchOS application to provide greater functionality to the user. The iOS application has a total of six UI screens, with the first four serving the same purpose as their analogous watchOS counterparts. However, to meet the goal of providing CPR session history to the user, the iOS application has two additional screens to serve this functionality.

#### 4.2.1. Saved Entries Screen

The first screen that is unique to the iOS application is the Saved Entries Screen. Put simply, this screen lists all of the saved CPR entries, organized by date and time - displaying each entry's date, time, number of compressions, number of breaths, and average pace. The user then selects which


Figure 11: Selection

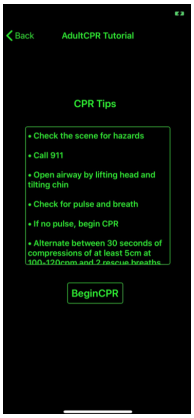


Figure 12: Tutorial

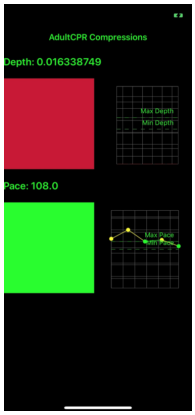


Figure 13: Comps

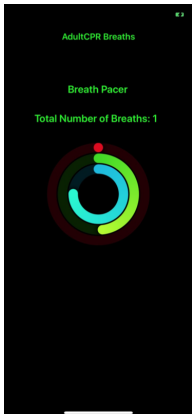


Figure 14: Breaths


Figure 15: Saved

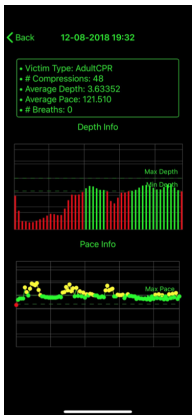


Figure 16: Details

specific entry they would like to examine, and are lead to the next screen with the appropriate information being forwarded. Conversely, if the user wants to delete a specific entry, this screen allows them to do so.

### 4.2.2. Detailed Report Screen

The second screen that is unique to the iOS application is the Detailed Report Screen. After the user selects a specific entry from the Saved Entries Screen, they are forwarded to this screen to better analyze that entry. In this screen, the user is presented with information of the victim type for that CPR session, the total number of compressions, the average compression depth, the average compression pace, and the total number of breaths. Additionally, graphs plotting the compression depth and pace are visible; in both graphs, the appropriate range is visible to the user, labeled with "Max" and "Min", so that the user knows how much of their data falls in this range. Additionally, the datapoints are color coded - the points that are below the correct range are red, those in the

appropriate range are green, and those that are greater than this range are yellow.

The iOS application is also implemented in Swift as it allowed for the use of various native libraries that allow for seamless communication with the watchOS application. The iOS application makes use of these key technologies to operate:

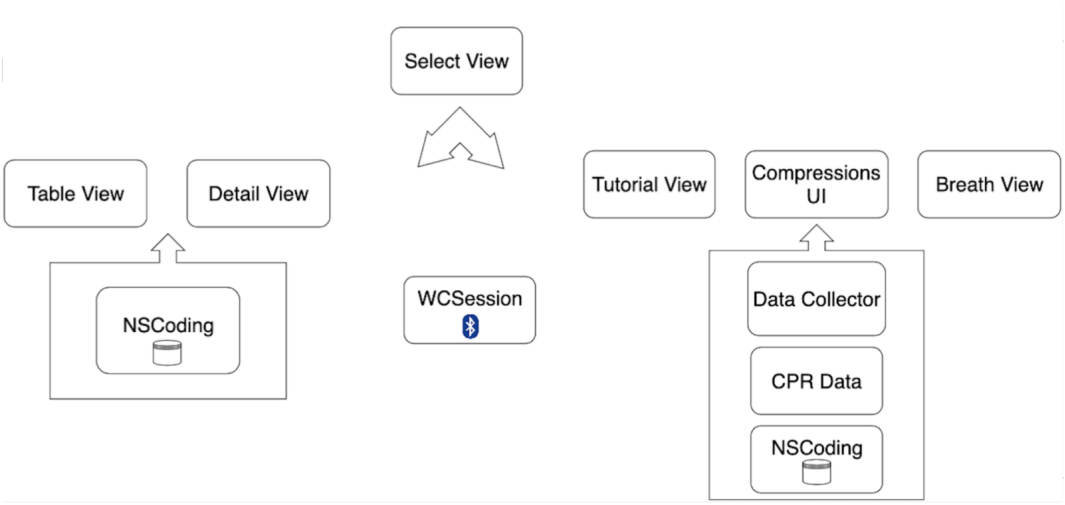


**Figure 17: iOS Application Architecture**

### 4.2.3. WCSession

Similar to its use in the watchOS application described above, the iOS application creates a *WCSession* instance to be able to communicate with the Apple Watch, but, in this system, it assumes a subservient role of receiving the various necessary data to keep the UI in sync and to collect the CPR session data.

### 4.2.4. Charts

The iOS application makes use of the popular iOS graphing library *Charts*, which is implemented in Swift. Using this library, the iOS application allows for the visualization of the collected CPR session data; specifically using a Bar Chart to display the compression depth data of the CPR session, and using a Line Chart to display the pace data of the session.

In addition to these standard technologies, the HeartWatch system relies on various proprietary iOS

solutions to function correctly as well. While there are many classes that the watchOS application uses, the most important are described below:

#### 4.2.5. CompressionController and BreathController

For every UI state of the watchOS app, there is a corresponding UI state in the iOS app that is kept in sync via messages from *WCSession*. These UI states are managed by their various controllers. All of these controllers are responsible for receiving and processing a message from *WCSession* to either update their own respective UIs or to transition to a different UI. The two controllers that are analogous to *BreathController* and *CompressionController* in the watchOS application also take care of logging the data that they receive using an instance of the *DataCollector* class.

#### 4.2.6. DataController

One instance of this class is created per CPR session. A reference to this class is passed between the various UI Controllers in the iOS application, and the appropriate CPR data is stored. For instance, *CompressionController* in the iOS application will use this *DataCollector* instance to store depth and pace data using the *addDepth()* and *addPace()* methods. Once the session ends, the *saveData()* method is called, using various *NSCoder* functions to save this CPR session data in a *CPRData* object.

#### 4.2.7. CPRData

*CPRData* is the data model used to store the various CPR session data. *CPRData* uses *NSCoder* functions to store:

- A double array of the depth values for every peak
- A double array to store the pace values at every peak
- An integer to store the total number of breaths administered
- A string to store the date that the CPR session was administered
- A string to store if the victim is an adult or a child

### 4.3. Data Collection and Processing

The key to the HeartWatch system is using the Apple Watch to get acceleration data, and then processing this data to get distance and compressions. The various techniques employed to process the data to get the most accurate values are described below:

#### 4.3.1. Singular Value Decomposition

No sensor is perfect, and therefore the raw values from the various Apple Watch sensors are not perfect - instead, the raw values are a combination of the real reading and a noise value. Therefore, one can represent any raw value as the sum of the real value and the noise:

$$RAW = R + N$$

The goal of denoising is to extract the real reading from the raw reading - using various methods that are appropriate for the type of data being processed. The technique employed to denoise the accelerometer is called Singular Value Decomposition, with its application to denoising accelerometer by the Q. Li research group [8]. This process is broken down into four distinct steps:

$$H_x = \begin{bmatrix} x_1 & x_2 & \cdots & x_k \\ x_2 & x_3 & \cdots & x_{k+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_m & x_{m+1} & \cdots & x_N \end{bmatrix}$$

**Figure 18: Hankel Matrix [8]**

$$H_x = U_x \Sigma_x V_x^T$$

**Figure 19: Singular Value Decomposition [8]**

1. The first step to denoising the accelerometer data is to construct a Hankel Matrix [8]. A Hankel matrix is a matrix such that each diagonal (going from the bottom left to the top right) has the same value. To construct such a matrix, one does so iteratively:
   - Assume a valid Hankel Matrix exists
   - Remove the first column of this matrix

- Copy the last column of the matrix. Remove the topmost element, shifting all of the other elements upwards.
- Then, at the bottom, add a new element.
- This inductive proof explains how to efficiently construct this matrix after the initial creation.

2. The second step for denoising accelerometer data is to perform a singular value decomposition on the constructed Hankel matrix of raw values. At a high level, the singular values of a matrix can be thought of as being "weighting factors" - with larger singular values corresponding to aspects of the data that are more important or significant [8]. Now, as explained above, the raw values in the Hankel matrix can be decomposed into a real value and a noise value. Therefore, applying this logic to the constructed matrix, one notes that since the noise values are one or two orders of magnitude smaller than the real values, then therefore smaller singular values will correspond to the noise, and larger ones to the real values [8].
3. The third step is to run a heuristic to zero all singular values below a threshold value [8].
4. The final step is to employ the various LAPACK operations to construct a new matrix via matrix multiplication, using these edited singular values (with the smallest ones set to zero). If the singular values were not edited, then the original matrix would be returned. However, because the insignificant singular values were zeroed, when the matrices are multiplied these removed singular values, which were related to the noise, contribute zero instead. Therefore, the final result is a matrix that is much closer to the true real values - effectively denoising the data [8].

#### 4.3.2. Transient Component Emphasis

Given that velocity and distance values do not come directly from a sensor, their values do not have the same amount of noise associated with them that the raw accelerometer values do. Instead, since both of these values are computed via integration, the biggest issue that arises in the data is the integration constant that results - as this constant can cause the values to be wildly inaccurate. Therefore, to remove the effects of this constant, a technique called Transient Component Emphasis is employed [12, 10]. This computation proves to be quite simple:

- Say that each raw integrated value is the sum of the real value and the integration constant:

$$RAW = R + C$$

- Therefore, subtracting two raw values results in the net cancellation of the integration constant - leaving only the delta between the real portions of the two raw values [**?**]:

$$RAW_2 - RAW_1 = (R_2 + C) - (R_1 + C) = R_2 - R_1$$

- Therefore, assuming the existence of an old value with the constant removed, a new value without the constant can be computed using this method; compute the difference between two raw values, thus leaving the change between their real components, and add this real change to the given old value - resulting in the desired new value [10, 12]:

$$NEW = OLD + (R_2 - R_1) = OLD + (RAW_2 - RAW_1)$$

## 5. Evaluation and Results

Initially, evaluation for the HeartWatch system was meant to be broken up into four distinct phases - compression depth evaluation, compression peak detection evaluation, CPR pace evaluation, and overall CPR improvement evaluation. However, various issues arose with the hardware that was to be used for evaluation, and therefore the evaluation techniques had to be amended. Therefore, below are the descriptions of what testing was able to be successfully done, what limitations arose due to the amended testing techniques, and what testing is still necessary to better evaluate the HeartWatch system.

### 5.1. Compression Depth Evaluation

Evaluation of HeartWatch's compression depth algorithm is done in two scenarios; one in an ideal, controlled scenario, and one "real world" evaluation - with each being described separately:

### 5.1.1. Ideal Scenario Evaluation

One manner in which HeartWatch evaluation is done is through an ideal-scenario, controlled test. This allowed for the evaluation of the algorithm without any noise or inaccuracies that would undoubtedly be introduced in a real world scenario, and therefore gave a way to evaluate the algorithm decoupled from the hardware specifics.

Unfortunately, such a device was not available for use when the HeartWatch system was being evaluated. Therefore, a more complicated and less accurate approach was chosen, with the steps being described below:

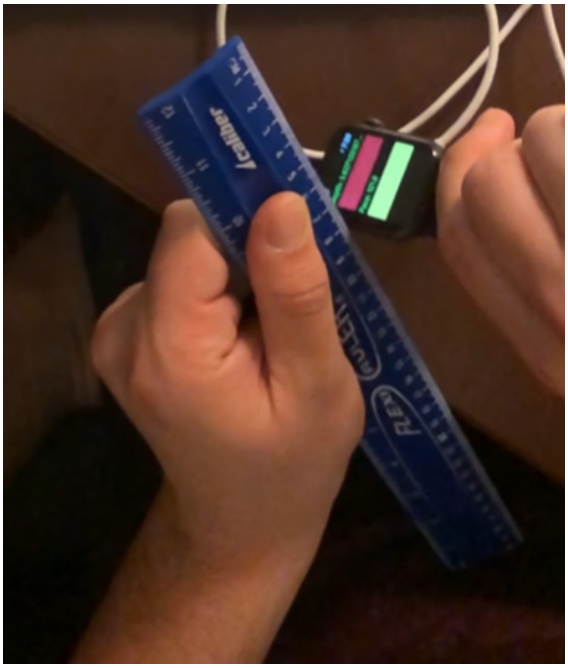

**Figure 20: Ideal Scenario Evaluation**

1. A calibrated ruler is held upright in one's non-dominant hand - ensuring that the ruler's tick marks are visible.
2. The Apple Watch is held in the dominant hand, ensuring that the screen is visible.
3. A CPR session is started on the Apple Watch
4. A slow-motion video is started, capturing the Apple Watch screen and the tick marks of the ruler
5. The test administrator then moves the Apple Watch forward and back along the ruler, attempting

to accelerate and decelerate in an appropriate manner and ensuring that both the Apple Watch screen and the tick marks are captured in the video

6. After 30 seconds of compressions (as this is the time that the Apple Watch switches to Breaths and therefore resets all of the compression values) were recorded, the session and the video capture are stopped
7. Then, the video is played back in slow motion, with the test administrator noting:
   - The administrator continues forwarding the video until a compression peak is reached - when the Apple Watch changes direction.
   - When a peak is found, the marking on the ruler is recorded, and the peak reading on the Apple Watch is also recorded
   - By computing the difference between the last ruler reading the the most recent one, the "actual" depth value is computed
   - This is compared to the value that is shown on the Apple Watch, and the difference is plotted
   - If the Apple Watch misses the detection of a peak, then this is noted as well

Employing this test framework, the following results are obtained:

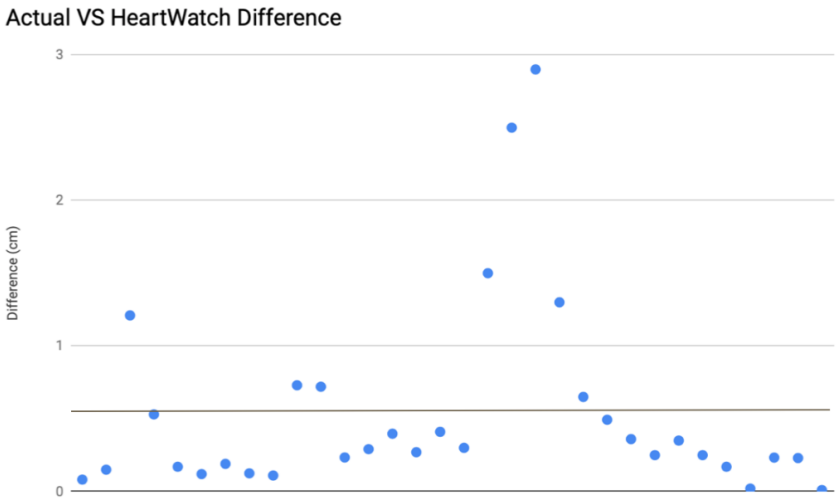


**Figure 21: Ideal Scenario Results**

| Ideal Scenario Results | |
|---|---|
| Mean Difference | 0.539cm |
| Standard Deviation | 0.673cm |

While these results are largely positive, note that there are two obvious outliers present that lead skew the mean and lead to a relatively large standard deviation. Once removed, the results prove to be considerably better:

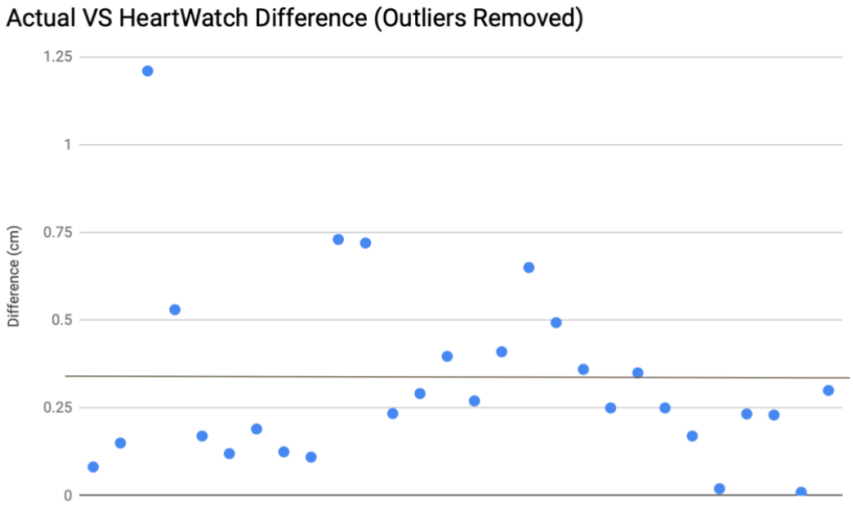


**Figure 22: Ideal Scenario Results (Outliers Removed)**

| Ideal Scenario Results (Outliers Removed) | |
|---|---|
| Mean Difference | 0.323cm |
| Standard Deviation | 0.259cm |

With the outliers removed, both the mean and the standard deviation are approximately halved - with the results being relatively close to that of the Song research group, which were a mean of 0.143cm and a standard deviation of 0.100cm [12].

### 5.1.2. Real World Evaluation

In addition to the ideal scenario testing, testing the HeartWatch system in a real world setting is appropriate, since no matter how accurate the algorithm proves to be in an ideal setting, if HeartWatch is not accurate when worn on one's wrist and used during real life compressions then it will not meet its goals of being a realtime aid. The two most notable differences that arise in

real world usage are: 1) When the Apple Watch is worn on the administrator's wrist, it is held at a different angle than in the controlled test. Additionally, being worn on the wrist allows for more lateral movement and twisting; the Apple Watch does not remain in a constant angle. Because accelerometer values are used to calculate distance, and these values are given as acceleration values in three axes, this rotating and twisting introduces the possibility of more noise and less accurate readings from the accelerometer - and therefore testing how HeartWatch performs under these circumstances is crucial to assess its usability. 2) When CPR is administered using a CPR dummy or on a live victim, the compressions are generally more forceful as the chest must actually compress. Therefore, in this real world usage one cannot expect the Apple Watch to be accelerating and decelerating in the same manner as it was during the controlled test, and as the accelerometer values are the key to calculating distance, it is important to test this scenario.

Obviously, one cannot safely use humans to evaluate CPR depth compressions for a real world test. Therefore, to evaluate the compression depth accuracy in a real world setting, a CPR mannequin, which simulates the resistance and feel of the average human chest, is employed. Just as one would do chest compressions on a victim's chest, one does chest compressions in the same fashion on the CPR dummy.

With such a mannequin, the manner in which the research groups evaluated their systems proved to be quite simple - a verified control device, like the CPRMeter 2, is obtained, and CPR is performed using both the control device and the system being tested - with their respective outputs being compared to determine the accuracy of the system under development [12, 13].

Unfortunately, since the CPRMeter 2 is considerably expensive, it was not available during the testing of the HeartWatch system, and therefore this test strategy was not able to be employed. However, all was not lost, since CPR dummies make an audible clicking noise when the threshold compression depth is reached. Therefore, one can evaluate CPR compression depth accuracy using the following technique:

1. Place the Apple Watch on the administrator's wrist
2. Start a CPR session using the HeartWatch system

3. Start a screen-recording on the accompanying iOS application
4. Begin doing compressions, ensuring to compress enough such that the dummy makes the audible click
5. Continue this for the appropriate 30 second interval
6. When this interval ends, play back the iOS application screen recording - comparing the values that are seen onscreen to the value that the dummy makes a noise at

**Figure 23: Watch on Top Hand**

**Figure 24: Watch on Bottom Hand**

This procedure is done in two different scenarios - one with the Apple Watch placed on the top hand, away from the chest, and another with the Apple Watch placed on the bottom hand, closer to the chest. This is done to ensure that the placement of the smartwatch is taken into consideration when testing the accuracy of the HeartWatch system. First, the watch on top results are reported:

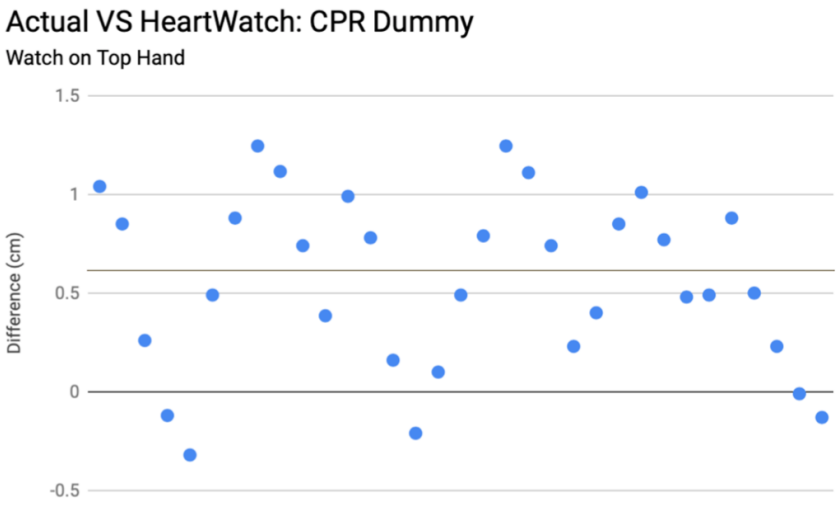


**Figure 25: Real World Results - Watch on Top Hand**

| Real World Scenario - Watch on Top | |
|---|---|
| Mean Difference | 0.559cm |
| Standard Deviation | 0.438cm |

As the results demonstrate, with the watch on the top hand - which is farther from the center of the chest, which is being compressed - the results are considerably worse than the ideal scenario proved possible. Noting that this may be due to the placement of the watch on the wrist that is farthest from the area being compressed, the same test is performed, but with the watch on the bottom wrist. The results are reported below:

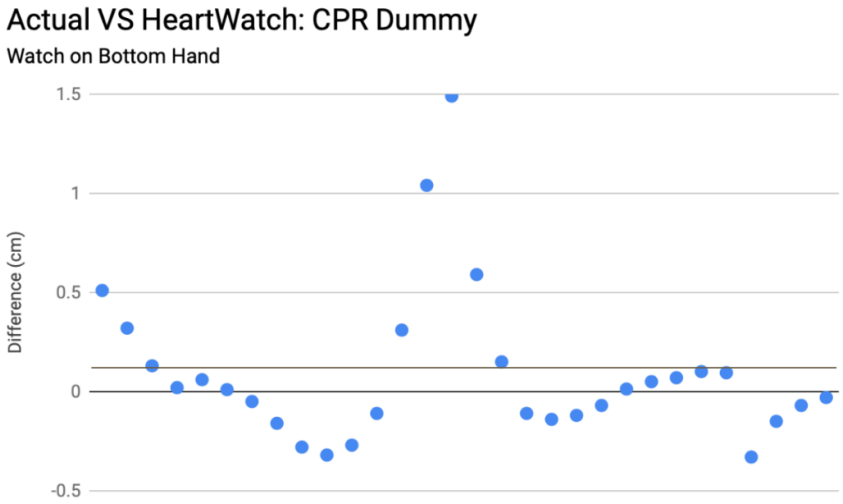


**Figure 26: Real World Results - Watch on Bottom Hand**

| Real World Scenario - Watch on Bottom | |
|---|---|
| Mean Difference | 0.091cm |
| Standard Deviation | 0.389cm |

With the watch on the bottom hand, the results are considerably better - proving to be about as accurate as the research group. This improvement switching from top to bottom hand makes sense; the bottom hand is the hand that initiates compressions and is closer to the center of the chest where the compressions are actually taking place, and therefore being placed on this hand is offers the HeartWatch system a more representative environment. However, one notes the presence of three outliers, whose removal demonstrate even better results:

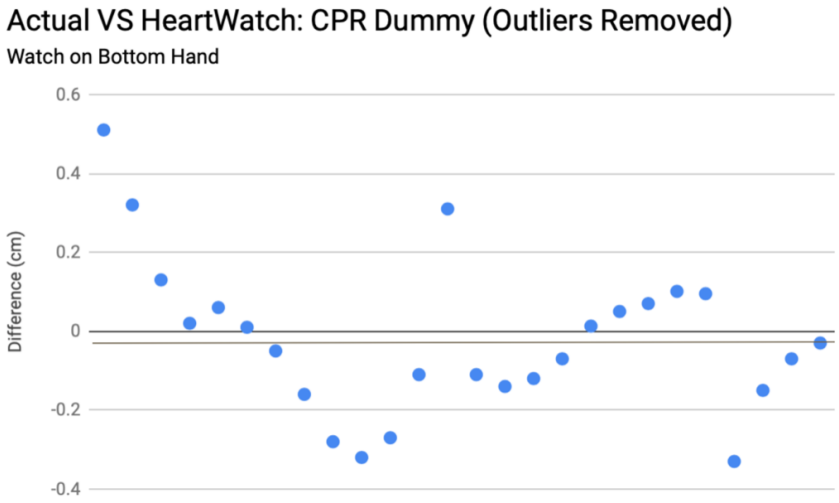


**Figure 27: Real World Results - Watch on Bottom (Outliers Removed)**

| Real World Scenario - Watch on Bottom (Outliers Removed) | |
|---|---|
| Mean Difference | 0.02cm |
| Standard Deviation | 0.198cm |

With the outliers removed, one notes that the mean is considerably better than that reported by the research group - demonstrating that the HeartWatch system, when tuned to remove such outliers, does well in its goal of providing realtime data to the user. While the results were largely positive, we note that in all instances the standard deviations were higher than that reported by the research group - this being attributed to two issues with this testing approach that should be noted: 1) The click is made the moment a threshold value is passed. Therefore, if one compresses past this value this information is not communicated by the clicking noise - this meaning that the administrator must be careful to stop compressing the moment that the click is heard. 2) Additionally, this implementation is sensitive to how precise the dummy is when it comes to making the audible click - if this click is made at different values, this approach becomes less accurate as well.

### 5.2. Peak Detection Evaluation

Key to the HeartWatch system is accurate peak detection, as a peak must be detected to know that a new chest compression is being initiated and that the previous compression has ended. Therefore, one notes that our depth and pace data is coupled to our peak detection algorithm - since if a peak is not accurately detected, then the compression that was performed is not recorded as having finished

- this therefore making it impossible to report to the user the depth of that compression in addition to it making it impossible to determine the rate of compressions.

Thankfully, the same experiments that were used for compression depth evaluation are capable of evaluating peak detection, since, in both ideal and real world testing scenarios, side-by-side videos of the HeartWatch system's screens and Apple Watch movement were captured - this allowing the test administrator to compare when the Apple Watch detects a peak and changes its UI to when a peak is actually reached. The provided graphs demonstrate when a peak is successfully detected, with gaps corresponding to at least one missed peak.

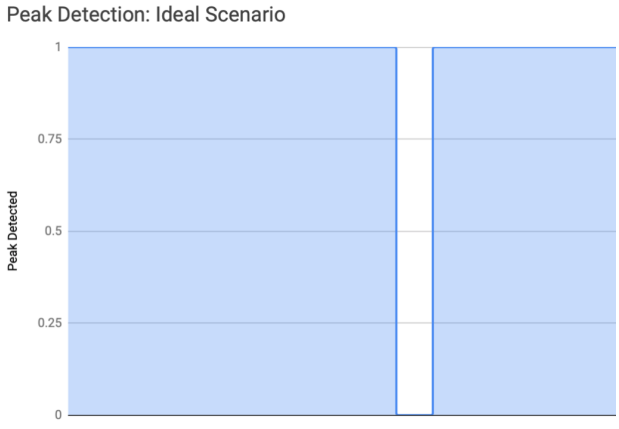


**Figure 28: Ideal Scenario Peak Detection**

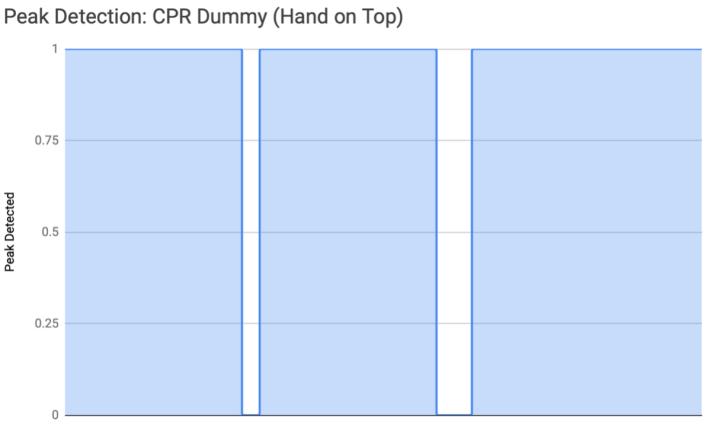


**Figure 29: Real Peak Detection - Watch on Top**

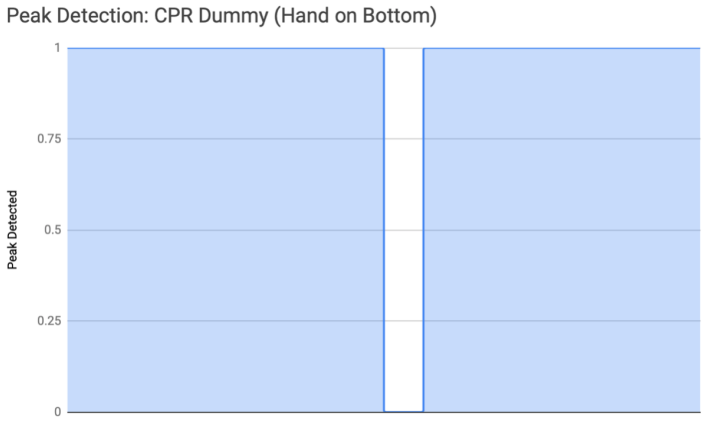


**Figure 30: Real Peak Detection - Watch on Bottom**

| Percentage Peak Detected | |
|---|---|
| Ideal Scenario | 94.3% |
| Real World - Watch on Top | 91.7% |
| Real World - Watch on Bottom | 93.8% |

In all three scenarios, the peak detection proved to be greater than 90%, which is quite good. Interestingly, for the Ideal and Watch on Bottom scenarios, the outliers that are present in the depth data occur right after the peaks that were not detected. While appearing like a coincidence, this makes sense - when peaks are missed being detected, the HeartWatch system continues to attribute the data being collected to the same compression even though a new one has been initiated - this therefore inflating the depth data recorded for the compression that the system thinks is still occurring. Therefore, while the peak detection is quite accurate, putting more development into optimizing it will prove to make the depth data more accurate as well.

### 5.3. Compression Pace Evaluation

The second part of HeartWatch's goal is to communicate realtime compression pace information, and this therefore must be accurately evaluated. The manner in which the pace is calculated is intrinsically tied to HeartWatch's peak detection - the calculated time between peaks is used to determine the current pace of compressions. Therefore, since compression pace is nothing more than a simple mathematical conversion given two compression peaks, no further testing was done for evaluating compression pace, as the pace results prove to be nothing more than the results from the peak detection evaluation. While one may still call for separate pace testing, such in depth testing would mean that peak detection testing would need to be considerably more complicated, and given the available tools and the time frame of this project, this was not able to be done.

### 5.4. Overall CPR Improvement Evaluation

The overall goal of HeartWatch is to help users better how they perform CPR in its totality. Therefore, assessing how using the HeartWatch system affects overall CPR administration - both for novices and for those that are trained in CPR - is a telling metric to assess.

Since the possible improvement of overall CPR administration is the goal, the most appropriate way to test this is using the given CPR mannequin; since this is the only way that a CPR administrator can practice actually performing CPR in its totality - on a dummy that mimics true CPR victims.

Unfortunately, the CPR dummy that was obtained proved to be especially dumb - the audible click from the mannequin was made at a value of about three centimeters instead of five centimeters, in addition to the fact that this click was made between a range of 2.5 through 3.5 centimeters. As this audible click is the only means available for the test administrators to determine if CPR is being done correctly - using the click to determine if the correct depth is reached, and using the times between clicks to determine the "true" pace - since this click was made at an incorrect value, it would not have been representative to use it to assess CPR administration as a whole.

Yes, one could have still used this dummy to try to get some data, but this would have included editing the HeartWatch system to offer feedback at a depth of three centimeters instead of the correct five - and therefore this test would not have been an indicator of if the administrators are compressing at the optimal depth. Additionally, since the pace of CPR is tied to the compression depth - since a larger depth means that the motion of compressions must be faster, as it travels a larger overall distance - this would have proven to not be representative for evaluating CPR pace as well.

Therefore, due to these limitations that were introduced by the faulty CPR mannequin, this style of testing was not able to be completed in the timeframe given for this project.

## 6. Conclusion

CPR is lifesaving technique employed by normal people and trained medical professionals world-wide due to its effectiveness in various medical emergencies. Unfortunately, a lack of knowledge of and training in CPR prevents many bystanders from administering this lifesaving procedure in situations were it could greatly benefit the victim.

Therefore, HeartWatch was developed to address this discrepancy. Using the combination of two popular consumer devices - an Apple Watch and an iPhone - the HeartWatch system succeeds in

demonstrating that accurate and beneficial realtime CPR feedback and aid can be delivered to users worldwide without their needing to invest in expensive and inconvenient hardware.

While HeartWatch proves that such a system is possible and helpful, certainly more research in this area should be done to improve upon the promising results demonstrated. Specifically, a more robust peak detection algorithm, with fewer errors, would prove to make the system even more accurate than it already proves to be - leading to fewer outliers in the data and this providing more consistent and helpful data to the user. Additionally, issues and lack of availability of more robust testing hardware made the testing of HeartWatch less precise, and therefore to get more accurate results a larger testing suite - with better testing hardware - would be beneficial.

The development of the HeartWatch system proved to be as frustrating as it was fulfilling and exciting. Having no prior experience in iOS development whatsoever, having to develop two separate applications - a watchOS and an iOS one - using Xcode and the Swift programming language proved to be a challenging and rewarding experience. Additionally, while the Apple Watch has a considerable amount of resources available, due to its small form factor and Apple's subsequent decisions to optimize its battery life, numerous unforeseen difficulties arose; specifically, keeping the watchOS application's timer running in the background proved to be quite difficult, with no tenable solutions for this being found online, causing me to develop a custom solution and post it for others to benefit from.

Coming from a background in lifeguarding, I truthfully believe that everyone should be well versed in CPR and basic first aid, as I believe that this will contribute to countless victims having their lives changed for the better. The HeartWatch system proves that society can move a step closer to this goal via realtime CPR feedback through the use of two common consumer products, and, with more research and development, maybe a consumer-ready version of this product will be acheived in the near future.

Lastly, I would like to thank my advisor, Professor Alan Kaplan, and our graduate TA, Molly Pan, in addition to the rest of Princeton's Computer Science staff for all of their help this semester - HeartWatch truly would not have been realized if it was not for their time and willingness to help.

# References


[1] A. H. Assocation, "Cpr statistics," in *CPR Statistics*, 2018.

[2] C. Clinic, "New cleveland clinic survey," in *New Cleveland Clinic Survey*, 2018.

[3] M. Faul, S. Aikman, and S. Sasser, "Bystander intervention prior to the arrival of emergency medical services," in *Bystander Intervention Prior to the Arrival of Emergency Medical Services*, 2016.

[4] S. L. Firm, "12 fascinating auto accident statistics," in *12 Fascinating Auto Accident Statistics*, 2016.

[5] L. Geddes and M. Boland, "Chest compression force of trained and untrained cpr rescuers," in *Chest Compression Force of Trained and Untrained CPR Rescuers*, 2007.

[6] C. C. HQ, "Avoid these mistakes when performing cpr," in *Avoid These Mistakes When Performing CPR*, 2015.

[7] J. Kolko, "Five thirty eight, how suburban are big american cities?" in *Five Thirty Eight, How Suburban Are Big American Cities?*, 2015.

[8] Q. Li, X. Chen, and W. Xu, "Noise reduction of accelerometer signal with singular value decomposition and savitzky-golay filter," in *Noise Reduction of Accelerometer Signal with Singular Value Decomposition and Savitzky-Golay Filter*, 2013.

[9] L. Medical, "Cprmeter 2," in *CPRMeter 2*, 2018.

[10] G. Milette and A. Stroud, "Professional android sensor programming," in *Professional Android Sensor Programming*, 2012.

[11] R. Ohmura and K. Yamamoto, "Feedback method evaluation of cpr support systems under practical situations," in *Feedback Method Evaluation of CPR Support Systems under Practical Situations*, 2016.

[12] Y. Song, J. Oh, and Y. Chee, "A new chest compression depth feedback algorithm for high quality cpr based on smartphone," in *A New Chest Compression Depth Feedback Algorithm For High Quality CPR Based on Smartphone*, 2015.

[13] Y. Song, J. Oh, and Y. Chee, "Smartwatches as chest compression feedback devices: A feasibility study," in *Smartwatches as Chest Compression Feedback Devices: A Feasibility Study*, 2016.

[14] H. Staff, "Cpr in adults: Positioning your hands for chest compressions," in *CPR in Adults: Positioning Your Hands for Chest Compressions*, 2017.

[15] S. Staff, "Cpr in a baby," in *CPR In a Baby*, 2016.