Reusing Ambient Light to Recognize Hand Gestures

Mahina-Diana A. Kaholokula
Dartmouth College

Reusing Ambient Light to Recognize Hand Gestures

Prepared by
Mahina-Diana Kaholokula

Abstract

In this paper, we explore the feasibility of reusing ambient light to recognize human gestures. We present GestureLite, a system that provides hand gesture detection and classification using the pre-existing light in a room. We observe that in an environment with a reasonably consistent lighting scheme, a given gesture will block some light rays and leave others unobstructed, resulting in the user casting a unique shadow pattern for that movement. GestureLite captures these unique shadow patterns using a small array of light sensors. Using standard machine learning techniques, GestureLite can learn these patterns and recognize new instances of specific gestures when the user performs them. We tested GestureLite using a 10-gesture dictionary in several real-world environments and found it achieves, on average, a gesture recognition accuracy of 98%.

Dartmouth Computer Science Technical Report TR2016-797
Submitted on: May 31, 2016

1 Introduction

Motivation. Recent years have witnessed a huge expansion in technological innovation and development. Trends like ubiquitous computing and the Internet of Things favor mobile devices and devices that can be embedded seamlessly and inconspicuously into their environments. In particular, this means that devices are getting smaller [16]. Smaller devices force the elimination of extraneous hardware (like keyboards and mouses) and simultaneously lead to smaller screen displays, if any at all. Today, we can see this in products like smart watches, portable tablets, and in a growing number of connected home appliances [4, 12, 19, 41]. As the field continues to change, we realize the need for alternative ways for the user to interact with these devices.

Gesture recognition, concerning the ability of a computer to recognize the body language of a user, is one possible solution to this problem. One of the key factors of a good user interface is familiarity, or intuitiveness [39, 5]. Since people already communicate at least in part with one another via gestures, gesture recognition is a natural choice for human-computer interaction as well. Gesture recognition also allows for a richer and more diverse language with which to communicate with our devices. Lastly, gesture languages provide comfort and freedom to the user due to the fact that they are often hands-free and can be used at a distance, as well.

There are already many applications for gesture recognition technology. Imagine living in a smart home where a swipe of the hand could turn on the lights, change the TV channel, or raise the temperature of the room. In an office space, the flick of a wrist could answer a phone call, scroll down webpages, draw a graph, or flick through a presentation. Environmentally, gesture recognition can help conserve energy in cases where manually unplugging unused electronics is difficult or too much of a hassle. In gaming, it can allow for greater user immersion through more detailed and varied interactions and has huge potential when we consider virtual reality games.

Research is already being conducted in this field with many promising results. Some of these explorations are using tools that are already widespread in daily life, like WiFi or audio-based systems [13, 20, 44, 57]. In this paper, we discuss the potential of using ambient light, another already-prevalent source, in gesture detection and recognition.

Existing Methods. Many researchers are already developing preliminary gesture recognition interfaces. The most common method is to use cameras to gather data that can be fed through image processing algorithms [10, 24, 35, 55]. For example, the Kinect sensor and the Leap Motion, which have been lauded for their precision, both use infrared cameras to detect hand movements and positions [47, 53]. However, camera-based approaches often cause privacy concerns for their users. Gupta *et al.* developed Soundwave, an example of an audio-based tracker which recognizes five gestures with 85-100% accuracy, but has difficulties filtering out ambient noise and finding pitches to generate and measure that won't bother children or pets [13]. Other devices, like the CyberGlove or the Wii remote, are created as wearables or hand-helds. These can generate lots of data from multiple sensors, like tilt sensors and accelerometers [21], but they can be cumbersome or inconvenient to wear. A more novel idea uses radio frequency (RF) signals to enable whole-home sensing. Wisee provides a proof-of-concept prototype that can detect nine gestures throughout a two-story home using two modified WiFi routers [44]. Although it's currently difficult to achieve a high granularity of gesture identification with RF signals, this field shows a lot of promise as a pratical and pervasive gesture recognition system. Lastly, light-based approaches to sensing have also been considered. LiSense, which uses shadows to reconstruct full 3D skeletons, is one such example [31]. However, LiSense requires an active control of the lighting framework, and the necessary modifications are not easily achieved in most environments. A more in depth discussion of these gesture recognition techniques takes place in Section 11.

Proposed Method. We propose GestureLite, a prototype system that performs gesture detection and recognition using ambient light. The sensing platform comprises of a 3x3 array of photodiodes, each hooked up to a single Arduino that captures all the sensor data and sends it to a standard laptop to be analyzed (see Figure 2).

GestureLite is based on the observation that, in an illuminated area, the hand will reflect and block light in a predetermined way, resulting in changing light intensities throughout the room. If the lighting in the room is fairly consistent, then each gesture corresponds to a unique change in the light intensities throughout the room. When a gesture is performed within the viewing field of the photodiode array, GestureLite will be able to record the new light intensities that result from the gesture and, using machine learning techniques, will classify the gesture in real-time.

First, GestureLite will need to collect roughly 10 training samples from every gesture that the user may want to use. Next, GestureLite fits the training data to a k-nearest neighbors model, where $k = 9$. Then, the product is ready for real-time use. GestureLite can automatically detect when a gesture has occurred and once the gesture has

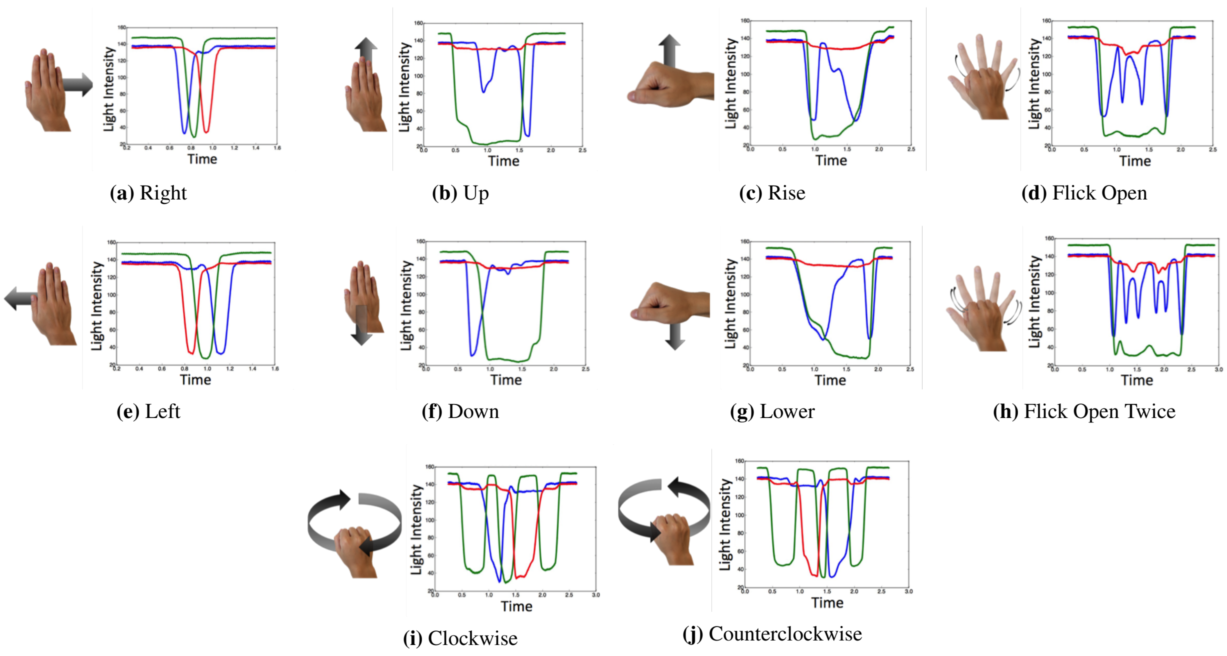


Figure 1: Gesture dictionary. Each gesture photo is accompanied by a graph depicting the shadow patterns created by that hand movement. The three lines correspond to three representative photodiodes from the nine-sensor array. All other line graphs in this paper also conform to this depiction.

finished, GestureLite immediately classifies it. We have implemented and tested GestureLite in multiple environments and our results reveal GestureLite correctly classifies about 98% of gestures. The 10 gestures that we use in this implementation are shown in Figure 1.

There are several advantages of using GestureLite as compared to other gesture recognition technologies. Most immediately, there are none of privacy concerns that a camera-based system might entail. It is essentially impossible to form anything more detailed than a silhouette from the photodiode information. The advantage of GestureLite over a wearable technology is that GestureLite is completely hands free, leading to more comfort for the user and less inconvenience over remembering to carry a device. Because GestureLite relies on ambient light, it is much easier and cheaper to install than a system like LiSense that needs full control over its environment. Lastly, GestureLite can be easily embedded in and adapted specifically to different environments. Gestures are thus able to have different meanings in different environments, unlike WiSee.

Challenges and Solutions. There were several challenges we had to resolve in order to make GestureLite performance-ready. The first challenge we came across when building GestureLite was how to deal with the abundance of sensor data that the system records. For a typical gesture, we can record up to 14,000 data points across the nine sensors. One of our earlier approaches to the problem was to use Dynamic Time Warping (DTW) to classify the gestures. This was promising because DTW can account for the variations within a gesture (challenge 2) without pre-processing the data. However, with such a large amount of data, this algorithm could take up to five seconds to classify a single gesture, which is not desirable for a system that is meant to be used in real-time. Instead, we decided to use KNN, a much quicker algorithm, as our classification technique. However, using the raw datasets in such a high dimensional space leads to other problems within KNN, so we also apply Linear Discriminant Analysis (LDA) to the data as a dimensionality reduction technique. This allows us to map the data from this huge dimensional space to a 10 dimensional space, which is much more efficient and appropriate to work with.

The second challenge was figuring out how to account for the small variations that occur within the same gesture. When the same gesture is performed twice, even by the same person, each instance will differ slightly in some aspect - perhaps speed or distance above the system the hand was held, for example (see Figure 4). These differences result in small changes in the light intensity that each photodiode registers, which affects our classifier's accuracy. There are also environmental variations to consider. As the sun moves throughout the day, the light contributed from any windows will change. These changes in light intensity result in slightly different shadow patterns for the same gesture, as well (see Figure 8b). To handle these variations, GestureLite performs a pre-processing step to normalize and standardize the data before applying the classifier. Environmental changes also make it difficult to properly detect the start and end of a gesture throughout the day, and this is accounted for with a periodic recalibration of the system.

A third challenge was dealing with a lack of knowledge about the position of the light source. A simple approach to this problem would be to create a "hit" order of how the shadows should hit the photodiodes when a specific movement is performed. However, because GestureLite is meant to work with ambient light, we could not rely on knowing which direction the light was coming from and without this knowledge, this geometric approach got very messy very quickly. The solution to this problem was to use machine learning techniques to capture the hidden patterns lying in the data.

Key Results. Final evaluation of the GestureLite system gives us the following key results:

- We prove the potential of ambient light-based hand gesture recognition. GestureLite classifies, on average, 97% and 99% of gestures correctly in the two tested environments, classroom and dorm room, respectively.

- We show that correct training samples representing multiple ambient lighting possibilities are important to our classifier's accuracy. By training data only collected at night and testing on data collected during the day, we see GestureLite's worst performance with an overall gesture recognition accuracy of 74% in the dorm room.

Contribution. With GestureLite, we make the following contributions:

- We propose the idea of using ambient visible light to affordably detect and recognize human hand gestures.

- We design a strategy to capture changing light intensities in the room and analyze them in real-time using machine learning techniques in order to classify each according to an established gesture dictionary.

- We build a proof-of-concept system using an array of nine photodiodes, an Arduino, and a laptop to perform the computations described above.

- We test our prototype in several indoor environments using a 10-gesture dictionary and evaluate the accuracy of our system.

Our work is one of the first to examine the feasibility of using ambient visible light for hand gesture recognition. We believe that because light is already pervasive in every indoor environment, it can be an easy and affordable solution to apply to HCI problems. We hope our work will inspire further explorations of how to re-use ambient light in other situations and applications.

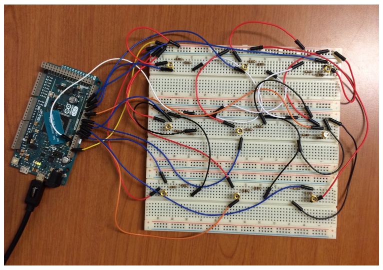


Figure 2: GestureLite Platform

2 GestureLite Platform

The GestureLite platform is built with nine photodiodes, three full-sized breadboards, and a single Arduino DUE. See Figure 2 for the sensing platform.

Photodiodes. Photodiodes are generally used to measure light intensity. Photons that get absorbed by the photodiode generate a current that can be measured; a larger current corresponds to a brighter illumination. While more precise technologies exist to measure the minute changes in the produced current, these products are much more expensive than the simple photodiode and Arduino combination that we use. Because GestureLite focuses on the overall pattern of a shadow and does not rely on exact values, a more precise measurement should not be necessary. We use Honeywell SD3421 Silicon PIN photodiodes with a 90° viewing angle in our system.

Affordable. The photodiodes we use in GestureLite are very low-cost (<\$2 wholesale). The three breadboards, resistors, and wiring are also commercially available at low prices (<\$10 total). The Arduino DUE itself is about

$50. In total, the system is fairly inexpensive to build and thus is reasonable for a commercial setting.

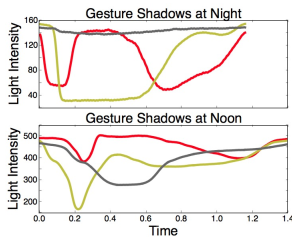
Arrangement. We connect three breadboards together to form the prototype surface. The photodiodes are arranged on the breadboards in a 3x3 array spanning 5-inches by 5-inches in total. I place two resistors (10MΩ) in series with each photodiode and connect the photodiode to power (5V) and the resistors to ground (0V). Each sensor is also connected to the analog input of the Arudino, which captures the currents produced by any light that hits the photodiodes. The Arduino itself is connected to a standard laptop that performs all the computations necessary for gesture recognition.

Capabilities. The photodiodes provide a wide field of view (90°) so that at any given moment during a gesture, at least one sensor should be under shadow. This is important in determining when a gesture has started and ended. The Arduino currently captures values from each sensor roughly every two milliseconds, which is enough for our prototype, but may need to be increased if the user plans on using extremely fast gestures. Lastly, the current array of sensors provides enough information for moving gestures, but fine-grained gestures (static gestures involving specific finger positions, for example) will require a more dense array of photodiodes.

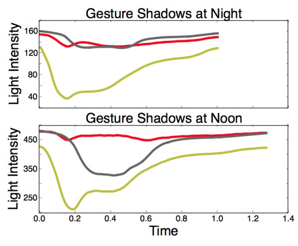
3 Preliminary Shadow Analysis

In this section, we note some observations regarding the shadows patterns that the hand gestures cast. We do this by studying the time-series graphs (the data from one sensor taken over the length of a gesture) that show the changes in light intensity caused by hand movements. First, we note that every gesture has a uniquely shaped shadow pattern graph compared to other gestures. This makes sense, as each gesture should "hit" each photodiode in a different order and for a different length of time depending on the speed, direction, and starting/ending point of the movement. Figure 1 shows the variations in light intensity for each gesture.

We also observe that shadow patterns change throughout the day. This occurs when ambient sunlight can filter through a window and brighten areas that were once in shadow and cast other photodiodes into deeper shadow relatively. An example of this is shown in Figure 3, where the *up* gesture is shown under sunlight in the bottom graph and shown with no ambient sunlight in the top graph. These clearly show that the same photodiodes detect different light intensities based on the time of day that the



(a) Dorm



(b) Classroom

Figure 3: Comparison of shadow patterns at night (top) and midday (bottom)

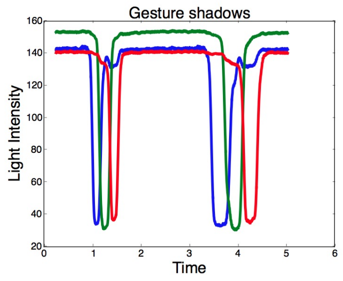


Figure 4: *Right* gesture performed twice

gestures were captured. Differences in shape, voltage, and magnitude are particularly evident.

Figure 4 shows one user performing the same gesture twice. This shows that even for a single user, each performance of a gesture will have some small variations, perhaps in speed, magnitude, or other properties. These distinctions can become more pronounced with the changes in ambient light discussed above. Thus, we realize we cannot use the raw data immediately for our KNN classification, which requires that all datasets of a given gesture be as similar as possible. To counteract these variations, pre-processing of the dataset is required before moving to the classification phase.

4 Assumptions

There are a couple assumptions and limitations that must be stated for this model.

Light Intensity. For one, the photodiodes will only capture current in a limited range of light intensity. In very bright areas (in direct sunlight, for example) the photodiode becomes saturated and the voltages recorded no longer reflect the light intensities accurately. Similarly, in very low light conditions, the differences in the light intensity due to a hand gesture become much smaller and so any patterns due to these changes will not be captured very well. These issues may be reduced by using different resistors in series with the photodiodes or using more powerful photodiodes.

Light Direction. It is obvious that the GestureLite platform and the lighting system should be fixed; GestureLite relies on creating shadow patterns due to a hand movement cutting off specific light rays from the lighting system to the photodiodes. If the system itself or the light source changes, we won't have the same shadow patterns being cast and recorded by the photodiode array. However, this does not mean that every light ray must be controlled; GestureLite is a fairly robust system and can handle small, natural variations within an environment, like that of changing sunlight streaming through a window.

Continuous Light Variation. GestureLite cannot deal with continuous changes in illumination (like the sun breaking in and out from behind clouds, for example). GestureLite needs to know the "standing" illumination of an environment (the light intensity when there are no user shadows) in order to correctly detect when a gesture has started and ended. Slow changes can be accounted for with a periodic recalibration, but continually changing intensities may move too fast for GestureLite to fully adjust.

Hand Placement. GestureLite only recognizes gestures that are performed above it (11above" referring to the space that directly blocks the photodiodes). Because the photodiodes have a limited viewing angle and precision, gestures that are performed far away and to the side of the system will not be registered. The impact of this issue could be lessened by using a larger array of photodiodes or using photodiodes with a larger field of view. However, this may also be a useful feature of the product if, for example, the user only wants to have gestures recognized for an activity performed in a specific location (at her desk in front of her computer, for example).

Gesture Speed. The user cannot move her hand too fast; a gesture ought to take at least a fifth of a second. This is a limitation from recording sensor values once every two milliseconds and also of the fact that we must standardize the time scale in order to apply our dimensionality reduction technique. Naturally, if we recorded more voltages per millisecond, we could accommodate faster gestures.

5 Procedure Overview

In this section, we cover the three stages of gesture recognition in GestureLite: gesture detection, data preprocessing, and gesture classification.

Gesture Detection. First, the system must recognize when a user is performing a gesture. There is a one-time calibration step to measure the typical voltage of a standing (unshadowed) system. A gesture is considered underway if any of the nine sensors return a voltage that is significantly under the average. The gesture is only processed if it occurred for longer than a fifth of a second; this is a necessary requirement for the dimensionality reduction step later on. It also helps eliminate any anomalies that crop up from random, extreme sensor readings. GestureLite also performs periodic recalibration to adapt to changes in the ambient light intensity.

Pre-processing. Gesture analysis occurs after the gesture has completed. The data collected from the sensors must go through pre-processing before recognition methods can be applied. First the data is normalized and scaled in both axes (magnitude and time), and then Linear Discriminant Analysis, a dimensionality reduction algorithm, is applied.

Classification. Classification of the gesture occurs last using machine learning techniques. Training data must be collected first (one time only) in the chosen location and throughout the time frame of planned use. To recognize gestures in real time, we use the k-Nearest Neighbors (KNN) technique, with $k = 9$. Extra measures are taken to try to ensure that non-gestures are not recognized as one of the 10 real gestures in the dictionary.

6 Gesture Detection

In this section, we discuss how GestureLite detects when a gesture is being performed. It is obvious that when a gesture is being performed at least one of the photodiodes should indicate a shadow passing over it. However, because there is inherent noise in the physical system hardware, it can be difficult to tell which dips in voltage are because a gesture is underway and which are only due to noise. For each sensor, GestureLite measures the typical variability of the data from a sensor in light, and when a voltage is recorded that lies outside this normal range of variation, we can assume that it is due to a real user gesture.

Calibration. A one-time calibration step must be performed when the system first turns on. The user must not cast any shadow over the system during this time, but otherwise does not need to interact with the system. Roughly five seconds of sensor data is collected from the unobstructed system. Then, for each sensor, the median absolute deviation (MAD) is calculated. The MAD is defined as the median of the absolute deviations from the data's median:

$$MAD = M_i(|x_i - M_j(x_j)|)$$

where x_j is the original data and M_i is the median of the series [30]. The MAD is a more robust measure of variability than using the standard deviation from the mean, especially with regards to outliers. This is crucial because we often observed that at least one photodiode reported an extreme voltage (> 1000) during the calibration step, and this heavily skewed the mean and thus the standard deviations of the time-series, resulting in poor gesture detection. The cutoff value of whether a low voltage reading is due to noise or a legitimate user shadow is at the median minus two absolute deviations. Figure 5 shows the cutoff values (dashed) for three representative sensors in GestureLite. From only this calibration step, GestureLite does not respond well to large changes in the brightness of the ambient lighting. The reason is that the cutoff values will no longer accurately describe the outliers in the current readings (ie. the values that must be from a human shadow). In this case, recalibration is necessary. Recalibration is discussed next.

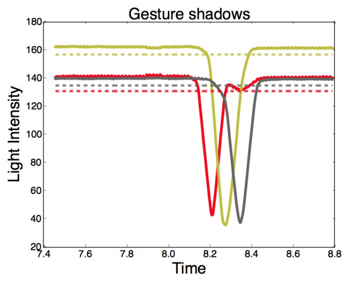


Figure 5: Cutoff values for gesture detection

Recalibration. Unless the user is in a completely static lighting environment (no windows, for example), the ambient light intensity will change over time. As the sun advances through the sky or moves slowly in and out of the clouds, the light intensities grow and wane. This makes it difficult to rely solely on the one-time calculated cutoff values for any long period of time. A brightening ambient light will result in gestures being detected late or possibly not at all. A darkening ambient light will be accidentally detected as the start of a gesture that never ends (unless the ambient light brightens again).

To solve this, GestureLite attempts to recalibrate the system periodically, if necessary. To check if the ambient light has dimmed, we check if a gesture has started but not ended. If so, we look at the median absolute deviation of the light intensities within this gesture data. Because our system is set to recognize dynamic gestures, any real gesture should show large variations for at least one sensor. In contrast, a "gesture" that is triggered by dimming light will not see these large variations. Thus, if GestureLite detects that the light variation is continuously under a given threshold for about 1.5 seconds, the system is treated as requiring recalibration. To handle a brightening atmosphere, GestureLite also calculates an upper cutoff value during the calibration step, defined at four absolute deviations above the median. If any of the sensors consistently register values above their respective high cutoff values, the system is treated as requiring recalibration. Both of these recalibration checks are performed roughly four times a second. GestureLite stores about five seconds of previous sensor data at all times; recalibration uses this historical data to compute new cutoff values in the same way as described above.

Start and End of Gesture. Every two milliseconds, GestureLite registers new voltages reflecting the light intensities seen by each photodiode in the sensor array. If any of the sensors have dipped below their cutoff points determined by the calibration step, GestureLite begins collecting the time-series data into arrays. Once all of the sensors report intensities that are back above their respective cutoff values, the gesture is determined to be over. Because the sensors have a wide field of view, there should never be a point in time where the hand is somewhere over the system without any sensors dipping below the cutoff. Lastly, any gesture that is shorter than a fifth of a second long is thrown out. Usually these very fast "gestures" are actually due to anomaly sensor readings that trigger past the cutoff points for a couple of milliseconds. The main reason for this elimination, however, is that in the pre-processing phase we average out sections of the data to arrive at 100 data points that represent the time-series. (It takes about $1/5$ seconds to record 100 values per sensor.) Finally, when all the data associated with a gesture is accumulated, it is passed on to the pre-processing phase.

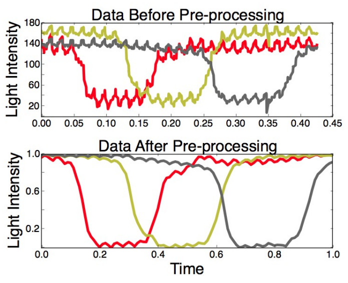


Figure 6: Time-series data before and after pre-processing

7 Pre-processing

We mentioned that the basis of GestureLite is recognizing a gesture from the particular pattern of shadows that it casts on the nine light sensors. However, every time a user performs a gesture, the time-series data from the photodiodes will be slightly different. There are multiple factors that could contribute to these small variations: 1) differences in human performance, like performing a gesture at a different speed or holding the hand at a slightly different angle; 2) slight differences in the light intensity, perhaps due to time of day or whether it's cloudy or sunny; or 3) variation due to inherent noise produced in the system. Each of these factors mean that we cannot apply classification techniques directly on the raw data. Instead, we must first pre-process the raw data to extract the core features that are better representations of the unique gesture pattern as a whole. Figure 6 shows the effect that pre-processing has on three representative sensor time-series. The top figure is the data without processing and the bottom figure shows the data after. Classification can then be applied to the post-processing dataset instead.

7.1 Normalization

We would like a gesture to be defined primarily by the direction of the hand movement. Direction is determined by the shapes of the graphs that each time-series creates. In order for the time-series to be comparable across gestures and to emphasize only the shape, the datasets must be adjusted so all values conform to the same scale.

Magnitude Scaling. The magnitudes of the currents will likely be variable within a gesture due to a variety of human and environmental factors, including how high the user gestures above the system and the general intensity of the ambient light. However, our classifier measures likenesses based on the exact voltages, which means that the absolute values of the data points matter. To fix this issue, GestureLite normalizes the magnitudes of all the gestures so that the greatest magnitude in a time-series is 1 and the lowest is 0. This is done by subtracting the minimum sensor value from all values in the time-series and then dividing by the max value remaining in the modified time-series. That is, for all x_i, with x being a time-series, we convert:

$$x_i = \widetilde{x_i}/MAX(\widetilde{x})$$

where

$$\widetilde{x_i} = x_i - x_{MIN}$$

Note that this does not change the shape of the time-series graph.

Time Scaling. The speed of a gesture can provide useful information. For example, an application may want to handle the same gesture slightly differently depending on the speed at which it's performed. However, varying hand speeds interfere with the ability of our classification method to process the data. This happens for two reasons:

1. We want to classify gestures based on how similar two gestures are at any given time in the gesture, where time is taken to be relative to the entire gesture, not an absolute value. This is equivalent to comparing the shapes of the shadow graphs- it matters that the minimum values from each sensor are occurring in the same order relative to each other, but it doesn't necessarily matter what the exact times are that the minimums occur at.

2. Our KNN implementation uses Euclidean distance to measure similarity, which means that every dataset associated with a gesture must be transformed into a single-dimensional vector in the same n-dimensional feature space.

Because it is physically impossible for a user to control the exact speed of any gesture, GestureLite must first scale and condense the time-axis of each time-series into the same single-vector n-dimensional space before continuing the rest of the classification process.

Creating a vector representation of the gesture data is easily addressed by flattening the nine time-series into one vector by joining them end-to-end. Now each element in the vector corresponds to a single sensor value at a single point of time. Before this, however, we want to standardize the length of each individual time-series so that our KNN implementation computes the distance between corresponding points in the shadow graphs (by relative, not absolute time). We observed that most gestures take at least 0.20 seconds long, equivalent to approximately 100 readings from each sensor. Based on this, we estimate that gesture patterns should be distinct and identifiable based on only about 100 readings. This is the length that we standardize each individual time series to. For each time-series, then, we divide the dataset into 100 equal chunks based on time. The values in each chunk are then averaged so that the resulting time-series is of length 100. This also has the effect of smoothing out the graphs, minimizing the effect of noise while still representing the same rough shape of its original shadow pattern graph. Only after this standardization do we join the nine time-series together to create a single vector associated with the gesture.

7.2 Dimensionality Reduction

In general, a gesture can take up to two seconds long, which corresponds to approximately 1,500 light readings from each photodiode. If we transform the nine sensor time-series into a single vector straight away, the resulting vector will live in an 14,000-dimensional feature space. With the time standardization process above, this can be reduced to a 900-dimensional vector. However, applying KNN within a 900-dimensional feature space is still a concern. There are two main problems with working in high dimensions: 1) it's inefficient and 2) our algorithm falls victim to the curse of dimensionality. Our KNN implementation uses Euclidean distance to determine similarity; obviously finding Euclidean distance in a high-dimensional space will be more complicated and less efficient than the same computation in a low-dimensional space. This is especially true because in high dimensions certain performance-improving measures cannot be taken [38]. These costs are amplified by the fact that the distance between the unknown gesture and *each* training sample must be found for KNN to work. However, the primary issue here is that in a high dimensional space, Euclidean distance becomes less meaningful. This is called the *curse of dimensionality*. Essentially this means that the differences in distance from the query point to any of the training samples gets smaller and smaller as the dimensionality of the feature space grows, rendering the concept of "nearest neighbor" meaningless [7].

Linear Discriminant Analysis. The solution to these problems is to reduce the dimensionality of the dataset even further using Linear Discriminant Analysis (LDA). LDA is a linear transformation technique that will project our 900-dimensional feature space onto a considerably smaller subspace while maximizing class-separability. In brief, the principle behind LDA is to minimize within-class scatter (S_w, the variance within a given class) and to maximize the between-class scatter (S_b, the variance between classes, roughly computed by using the mean vectors of each class). The transformation that will maximize the ratio of between-class scatter to within-class scatter is computed by finding the eigenvectors that correspond to the dominant eigenvalues in the matrix given by $S_w^{-1} S_b$ [6].

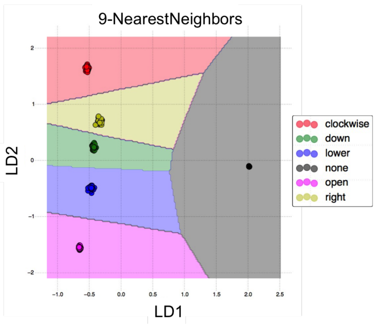


Figure 7: Example of LDA and KNN. LDA projects samples from six gestures onto a 2d space that maximizes distance between classes. When a new hand movement is detected, it is transformed into the same 2d space, and then classified by its nearest nine neighbors (colored here).

GestureLite uses LDA as following: 1) the training data is used to create the transformation matrix, T, that maximizes class separability; 2) T is applied to the training data, reducing each sample's dimensionality from 900 to 10; 3) our KNN model trains on these transformed samples; 4) as new gestures are captured, T is applied to the new dataset and the resulting 10-dimensional vector is classified according to its nearest neighbors in the model (see Figure 7).

8 K-Nearest Neighbors

As we have mentioned, GestureLite uses k-Nearest Neighbors to classify gestures. KNN employs lazy learning, which means that the model does not generalize beyond the training data until a query is made. GestureLite builds the KNN model using pre-processed training samples that have been recorded by the user. The user should record about 10 samples of each gesture in the various possible lighting scenes that could occur. Additionally, 10 samples of the standing scene (the system under unobstructed light) should be obtained as well. This is one way to try and handle false gestures, in this case the small fluctuations in light intensity that may occur naturally in the environment or any movements that the user may make mistakenly in partial view of the system.

Classification. KNN is one of the simplest machine learning algorithms. Anytime a new gesture is captured, the dataset goes through the pre-processing steps detailed in the previous section and then is classified according to the majority class of its k nearest neighbors, 11nearness" being determined by the Euclidean distance between the samples [37]. GestureLite sets k to be 9. We find that KNN works extremely well despite being so simple; this is in part due to the large amount of pre-processing that the time-series go through first (especially the LDA which separates the classes further), and also because as a largely single-user system, there is less likely to be strong variation within a gesture class. Thus we expect that any gesture the user makes will be "close" to other instances of the same gesture and farther from all the others.

While GestureLite shows a strong accuracy regarding the classification of "correct" gestures (ie. belonging to a class in the gesture dictionary), it struggles to point out gestures that are of an unknown (untrained) class. One method we use to weed out false gestures, mentioned above, is to train a *none* gesture to recognize a mostly open system. This protects against GestureLite attempting to classify shadows that mistakenly hit a part of the system or light shadows from background/ far away movement. Because KNN only knows how to assign a new sample to its most similar class, there isn't much opportunity to check whether the new gesture might not belong to any class at all.

9 Evaluation

To evaluate the performance of GestureLite, we collected roughly 140 and 160 samples of each of the ten gestures in two locations, respectively. The first is a dorm room with a single overhead light and next to a large window, and the second is a classroom with multiple overhead lights and small windows along a sidewall. All samples were taken by one user. The test user did not collect more than 10 samples in one sitting to better simulate the small variations that are likely (even for a single user) within any given gesture. The samples were collected over 4-5 days during different times of the day to capture the changes in ambient light that occur throughout the day. In total, we have 1,547 dorm room samples and 1,753 classroom samples that range from being taken at night (no ambient sunlight) to high noon (maximum ambient light filtering in through windows).

9.1 Overall Performance

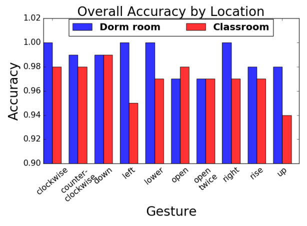
We evaluate the overall accuracy of GestureLite using a 16-fold stratified cross validation on all the samples collected in a each location. This corresponds to randomly partitioning all the data from each location into 16 equal-sized sections. Of the 16 sections, one section is used for training and the rest are used for validation testing; this is repeated 16 times using each of the of the sections as the training data once. A 16-fold stratified cross validation allows for roughly 10 samples from each class per partition. During this time, we use a k-value of 9 in our K-Nearest Neighbors classifier. These decisions are explained in Section 9.4.

Figure 8a shows the fraction of times that each gesture was correctly classified in the 16-fold cross validation. The blue bars display the results from data collected in the dorm room and the red is from data collected in the classroom. In the dorm room, we observe that the overall accuracy for recognizing gestures in the established dictionary is 99%, with the lowest being 97% for two gestures, *flick open* and *flick open twice*. In the classroom, we observe that the overall accuracy for gesture recognition is 97%, with the lowest being 94% for the *up* gesture and the highest at 99% for the *down* gesture.

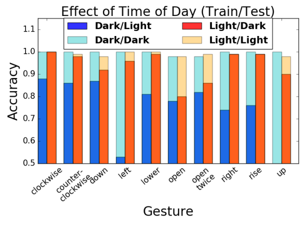
9.2 Ambient Light

We evaluate the effect that changing light intensity has on recognition accuracy. Light intensity changes due to extra light that may be filtered through the windows. (See Figure 3 for a visual aid of how shadow patterns change throughout the day.) This evaluation consists of training solely with data collected at night and then testing on data collected during midday. The reverse is also tested. We do a k-folds evaluation again, separating the training data into samples of size 10. For each training sample, we validate using all the test data available for that group. Data collected during low-light periods (sunrise and sunset) was removed for this evaluation in order to maximally demonstrate the effect that changing ambient light can have on gesture recognition. Evaluation was done on dorm room data where ambient light from the window had a bigger effect.

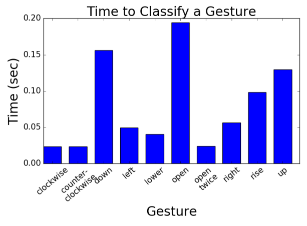
Figure 8b shows the results of this evaluation. Each gesture has four bars associated with it. The dark blue bars show the accuracy of recognition when training on night data and testing on midday data. The green bars show accuracy when both training and testing is done with night samples. The red and yellow bars are the reverse of these tests (with regards to midday and night data), respectively. The results show that changing ambient light



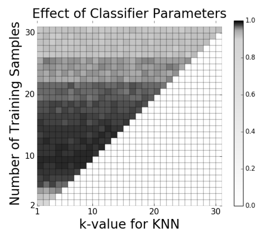
(a) Overall Accuracy



(b) Time of Day Comparison



(c) Recognition Latency



(d) Classifier Parameters

Figure 8: Testing

can have a serious effect on the ability of the classifier to identify the gestures. Training and testing data within the same light intensity resulted in an average recognition accuracy of 99% and 100% and lows of 97% and 98% for the nighttime and midday groups, respectively. We see these numbers drop quite a bit when mixing up the training and testing samples. Training on midday samples and testing on night still allows for a reasonable accuracy of recognition at 95%, although the low drops down to 80% for the gesture *flick open*. Training on night samples and testing on midday, for some gestures, make the system unusable. The average accuracy across all gestures is 74%, with *up* and *left* having the lowest individual accuracies of 30% and 53%, respectively.

9.3 Recognition Latency

One of our main goals with GestureLite was to create a system that could be used in real time. We evaluate the average (mean) length of time it takes to process and classify a gesture, starting from the moment the gesture ends. We used over 100 gesture samples in the dorm room environment. This was done using 10 training samples per gesture and $k = 9$.

Averaging amongst all gestures, GestureLite takes roughly 0.08 seconds to classify. Figure 8c shows the average number of seconds it takes to identify each individual gesture, with seven gestures taking under a tenth of a second to classify, and the maximum (belonging to the *flick open* gesture) taking just under a fifth of a second, on average.

9.4 Classifier Parameters

As for most machine learning algorithms, there are classifier parameters to establish. In our case, we have two: 1) the size of the training dataset and 2) the k-value of our KNN classifier. In this section, we evaluate how the size of the training dataset (number of samples per gesture) affects the recognition accuracy and how the k-value affects recognition accuracy. Because these two are related (a bigger training sample likely requires a bigger k-value to avoid overfitting), we evaluate them together. For each k-value between 1 and 30 and for each sample size between 2 and 30 we perform k-fold cross validation on the entire dataset to compute an overall gesture recognition accuracy for our established dictionary. It makes little sense to have a k-value that is larger than the training sample size, so we do not analyze the performance of these value pairs.

Figure 8d shows the results of these tests. (These are results from the dorm room data, but the classroom data shows similar results.) The x-axis shows possible k-values and the y-axis shows the training set sizes. In this map, the colors reflect the overall accuracy of the classifier, with darker colors corresponding to higher accuracies. In general, we see that the optimal sample sizes are between 8-11 when used with k-values between 7-10. Accuracy in this area is around 98%. Small sample sizes (<5) and big sample sizes (>25) see lower performance, classifying about 90% of testing samples correctly. Based on these results, we perform our tests using a sample size of 10 and a k-value of 9.

10 Future Work

While GestureLite works well with a 98% recognition accuracy in the environments laid out above, there is still a great deal of room for more research in this area.

Range of Vision. GestureLite has a fairly limited field of vision and the hand must be inside this area for gestures to be detected. A larger array of photodiodes or using photodiodes with a wider field of vision could increase the area of detection.

Light Saturation. GestureLite does not work under very bright lights or in the dark. In bright lights the photodiodes become saturated, while in darkness, shadows are not well defined for the sensors to pick up. These may be, to a certain extent, unavoidable problems when relying solely on ambient light.

Shadow Interference. Extra shadows in the environment, due to the user's body or other people, interfere with GestureLite's performance. Also, as previously mentioned, GestureLite cannot function under consistently shifting light intensities. These are all areas for improvement if GestureLite is to become a more stable, user-friendly product.

User Generalization. Currently, GestureLite has been tested as a single-user system. In the future, we would like GestureLite to recognize gestures from any user. This will require more research into how similarly different people perform the same gestures. It would also be interesting to try and allow multiple users to perform gestures simultaneously.

Gesture Granularity. GestureLite currently works for a pre-established dictionary of dynamic hand gestures. We would like GestureLite to recognize finer-grained gestures (using the fingers), which will likely require a denser array of photodiodes. We would also like to recognize static hand positions, which would require a new method of detecting when a gesture starts and ends.

Hand Tracking. GestureLite currently only performs gesture classification. The disadvantage of classification over full hand reconstruction is that the language the device recognizes is limited to the pre-defined dictionary. Furthermore, there is a learning curve to conquer before interaction between user and machine is smooth. Expanding GestureLite to be able to track hands and fingers in real time using ambient light would be a huge advancement in hand reconstruction. Perhaps a first step in this direction is allowing GestureLite to recognize when multiple gestures are performed consecutively, which will require a new method of recognizing when a gesture starts and ends.

Photodiode Arrangement. We set up the 3x3 arrangement of photodiodes to record the largest range of data relevant to the hand movements. This does not mean that this arrangement or array size is optimal. Figure 9 shows the overall accuracy of GestureLite (computed with a cross-validation test as in Section 9.1) when we only use the data from a couple of the sensors. These figures prove that all nine sensors are not necessary for accurate recognition. It would be useful to know what arrangement of sensors and what size array is required for an ambient light-based recognition system to work.

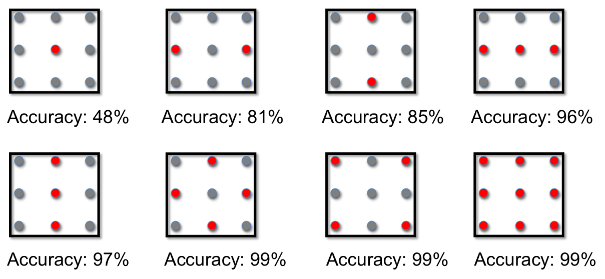


Figure 9: Gesture recognition accuracy based on different photodiode arrangements. The red circles depict the sensors whose data was actually used for the accuracy evaluation.

11 Related Work

In this section, we discuss related research on gesture recognition technology. There are two basic strategies to approaching hand gesture classification. One is a geometric approach, which reconstructs the hand based on certain movement and position constraints of the hand [28, 33]. This approach requires no training and is easily extendable to any number of gestures. However, it can be computationally expensive to fit the hand model (which has 27 degrees of freedom), and, depending on how the data on the hand is collected, this approach may result in low accuracy recognition. The second is a machine learning approach, where the computer is taught what each gesture should look like [10, 35, 47, 55]. This approach, while requiring possibly many training samples , can also be faster to run and much easier to implement. Because machine learning is based on training samples, this approach is less generalizable to an entire population of users. A discussion on current tested approaches of gathering data on the hand follows.

Vision-based. The camera-based approach is perhaps the most common approach to gesture recognition. The simplest technique uses a 2d image or video stream input of the hand [10, 24, 35, 55]. Image processing, often through skin color analysis, motion analysis, and edge detection, is used to identify and separate the hand region in photo. However, image processing often leads to constraints on the surrounding environment, as skin color information may be distorted by poor lighting or hard to distinguish against a similar color background. Furthermore, there are problems of occlusion that always occur when mapping a 3d shape down to two dimensions; essentially every 3d hand position will become a 2d silhouette. Lastly, using a camera to gather data is not a pervasive solution; the hand must be in the field of view of the camera for detection to take place.

To try to retain more 3d information about the hand's position, it's possible to splice together information from multiple 2d cameras [48] or use stereo cameras for a depth image [17, 18]. However, these usually require a much larger and more expensive setup; Sridhar and Sowmya needed four machines and 12 cameras for their system [48]. Limitations for these systems have to do with algorithm efficiency. Multiple cameras will use more processing time and reliable real-time stereo algorithms are not easily obtained or implemented [35].

Infrared cameras are also being used in gesture recognition systems, now [2, 8, 47]. Breuer's infrared time-of-flight cameras produce a low frequency light field which is reflected off the environment and sensed by the camera. Distances between the camera and the objects in the environment can be computed by measuring the time the light takes be reflected back to the camera sensor [8]. Using infrared light thus solves two of the problems facing the normal 2d cameras: 1) infrared cameras do not rely on skin color analysis and so are more robust against lighting changes in the environment and 2) infrared cameras can measure distances which means they can work better with occluded objects. In the case of hand recognition, this means that gestures with fingers or hands placed one in front of the other can still be recognized. The Microsoft Kinect [40] and the Leap Motion [26] are both examples of systems that can perform precise hand detection and recognition [15, 23, 47, 53]. One downside to IR cameras is that they require special equipment (IR emitters and sensors), unlike the 2d cameras that are already found on almost any laptop or desktop computer today. Finally, any camera-based approach will trigger privacy concerns, especially those that require storing images of possibly hundreds of training samples made by the user.

Audio-based. Although a more sparsely researched area, Gupta *et al.* have shown that a sonic-based gesture recognition system can exist and perform well. Their system, Soundwave, leverages a device's speakers and microphones and uses the Doppler effect to detect motion around the device. This detected motion can provide enough data to perform gesture recognition around the device. An audio-based system like Soundwave is particularly promising because it can detect gestures without a line-of-sight. However, work on Soundwave is still required to find a pitch that isn't audible to us (children or pets are particularly sensitive to high pitches). Finally, because the Doppler effect depends on movement, this method only works for dynamic, not static, gestures [13].

Wearables and Hand-helds. The rise of the smartwatch is a testament to the potential that wearable technology has in todays world. Wearables are so well liked by researchers because they offer a platform for many different types of sensors to occupy simultaneously. The CyberGlove III, for example, has over 20 sensors that can capture minute hand and finger movements [11, 21]. These sensors provide intimate details that can lead to accurate gesture recognition systems. Lu *et al.* developed an armband that can recognize 19 gestures with 95% accuracy using electromyography (muscle) sensors and an accelerometer [36]. Xu *et al.* created a classifier using a smartwatch equipped with an accelerometer and gyroscope that identifies finger writing and 37 distinct gestures with 95% and 97% accuracy, respectively [54]. Wearables also have fewer location constraints; they are portable and users don't need to be concerned with blocking sensor signals (versus camera-based systems, which must have a straight line of sight between camera and hand). For example, Nijron *et al.* developed a ring for users to wear that allows them to type on an imaginary keyboard anywhere with a flat surface [43]. However, wearables have a number of disadvantages, too. Their primary drawback is that users do not have free, unimpeded immersion with their device. Instead, they must remember to put on and take off this second device that may also be cumbersome or uncomfortable to wear. Also, because they are physical items, wearable technology is more likely to get damaged than the fully hands-free approaches.

Similar to wearables, there are also separate hand-held controllers that are supposed to act as extensions of the arm and can also be used for gesture recognition and pointing. The Wii remote [45, 42] is an example of this and it has the same advantages and limitations as those above, although it is perhaps even more cumbersome to use since it must be actively held at all times.

RF Signal-based. A very promising new modality for human gesture recognition involves using radio frequency (RF) signals. These signals are already pervasive in mobile phones, televisions and other wireless devices, allowing this method to be easily assimilated into our everyday use. Most research in this field is based around the idea that RF signals become distorted in the presence of hand movements, and these amplitude variations are distinct enough to identify different gestures. AllSee [20] and SideSwipe [57] are two examples based on this theory, using TV/ RFID transmissions and GSM signals from mobile phones, respectively. Another advantage of using RF signals for hand gesture recognition is that a line-of-sight is not required. WiSee, created at the University of Washington, uses standard WiFi to perform full-body gesture recognition with 94% accuracy throughout the home and through walls [44]. So far, there has been little research on more granular gesture recognition and single point tracking. However, Sun *et al.*'s WiDraw demonstrate that hand motion tracking can be done based on the power of WiFi signals [51].

Light-based. Visible light is a powerful tool for any application because it is so prevalent in our daily lives. For the most part, visible light communications (VLC) studies have been focused on wireless communication through modulating LED light intensities [34, 46]. Less research has been done on using light as a basis for gesture recognition, and any that has performs poorly in the presence of changing ambient light. Okuli, a prototype system made by Zhang et al., is one system that uses light to perform finger tracking. It uses a low-power LED and two photodiodes to locate the finger based on how the light is reflected off the finger. While a promising step in the right direction, Okuli can become unstable due to interference from ambient lighting or other reflective surfaces in the environment [56]. LiSense, a work very similar to ours, reconstructs the full 3d human skeleton of a user based on an analysis of the shadows they cast. The LiSense testbed is made of a large array of photodiodes on the floor and a similar array of LEDs on the ceiling [31]. Because LiSense is created for large-scale skeleton reconstruction, it does not have the ability to accurately recognize hand and finger gestures. Furthermore, LiSense requires full control of the lighting environment to function properly, which is not a feasible option for most commercial buildings.

12 Applications

There is a wealth of opportunity in the practical applications of human gesture recognition. In this section, we discuss a few possible applications and the research currently being done in these fields.

Man-machine Interfaces. Gesture recognition interfaces could support, if not completely replace, the current physical hardware that is necessary for user-machine interactions. Gestures are already a part of daily human-to-human communication; they would be a natural choice for replacing the keyboard and mouse of a desktop computer [35]. As devices become more portable in an age of ubiquitous computing, they inevitably become smaller. A product like the TypingRing [43] allows users to easily type on an imaginary full-sized keyboard rather than squint at the miniature keyboard on a mobile phone. In fact, Apple has recently filed a patent which details the use of photodiodes and LEDs to create hover-sensing displays that augment the traditional keyboard and trackpad [9]. As the IoT continues to amass interest, smart homes, filled with smart devices, will become popular. Being able to control smart home systems and devices from anywhere in the home, hands-free, is possible with gesture recognition [29, 44].

Graphic Editors and Visualization. Graphic editors could benefit from hand gesture recognition or tracking systems. Drawings and animations, especially in 3d, are much more intuitive to create with the hands than with a standard mouse. KinectPaint [25] and WiDraw [51] are a couple examples of how hand and finger movements can be tracked and used to create drawings on a computer. Gesture recognition can also be applied to visualization technologies, allowing for easy manipulation of viewpoints and zoom functions of 2d or 3d images. This is relevant to any number of jobs, from architecture to fashion design to video special effects editors to medicine.

Computer Games and Virtual Reality. Using gestures in computer games and virtual reality (VR) creates a more immersive experience. Hong *et al.* have developed a system to play a Chinese chess game based on hand gestures [27]. Ghyme *et al.* have built a 2d computer game where an avatar is controlled (through movements and interactions) via hand gestures. One particularly exciting application is based on a study that shows that playing video games can help stroke victims regain motor control and range of motion [3]. Gesture-based video games can makes these games both more fun and also can take an active role in helping rehabilitate people with physical injuries. The most important aspect in virtual reality, perhaps, is immersion. Gesture recognition and tracking use in VR can help move toward a completely immersive experience. A user can point in the direction they want to move or use gestures to interact with objects in the virtual world [50].

Robotics Control. Other research has been focused on how gestures can be used to operate robots remotely. There are multiple studies concerning having a robot arm replicate the motions of a user, including the ability to pick up and put down objects [1, 22]. Waldherr *et al.* has developed a robot that picks up trash at the specific locations pointed to by a human user [52]. Robotics control has various applications in areas like surgery, construction, and research projects in extreme environments, to name a few.

Sign Language. A popular goal of gesture recognition research is to accurately and wholly recognize sign language [14, 49]. Sign language is the common mode of communication for speech and hearing impaired people. However, the majority of people do not understand it. Thus, a sign language gesture-to-speech interpreter could be a huge aide in crossing this language barrier. A sign language recognition system could also help otherwise untrained people learn the language easier. Promising advancements in the field have been made, with a system made by Liang and Ouhyoung recognizing sentences made from a 250-word vocabulary with an accuracy around 80% [32].

13 Conclusion

In this paper, we designed, implemented and tested GestureLite, a system that provides hand gesture detection and classification using ambient light. We observe that every hand movement casts a unique shadow pattern over the GestureLite platform. GestureLite processes the light intensity data gathered from the physical platform and extracts the core features of these patterns to apply them to a KNN algorithm for classification. Our evaluations show that, with proper training samples taken throughout the day, GestureLite can accurately classify 98% of user gestures. GestureLite is a low-cost product that is adaptable to any indoor environment, making it a strong option for commercial use. Already, the applications for gesture recognition systems like GestureLite are endless. GestureLite can be used in homes to interact with smart appliances, or manufacturers can embed photodiodes into tablets and phones to augment their touch screen capabilities. We hope that our work will inspire more research and innovation in both the gesture recognition field and in the possibilities of re-using ambient light as a medium for obtaining and communicating user information.

References

[1] L. Aggarwal, V. Gaur, and P. Verma. Design and implementation of a wireless gesture controlled robotic arm with vision. *International Journal of Computer Applications*, 79(13), Oct 2013.

[2] Y.-K. Ahn, K.-S. Choi, Y.-C. Park, and K.-M. Jung. Infrared camera-based hand gesture space touch system implementation of smart device environment. *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 7(10):1318 – 1322, 2013.

[3] G. Alankus, R. Proffitt, C. Kelleher, and J. Engsberg. Stroke therapy through motion-based games: A case study. In *Proceedings of the 12th International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '10, pages 219–226. ACM, 2010.

[4] Apple Inc. Apple ipad. Available at `http://www.apple.com/ipad/`.

[5] Argon Design. 5 aspects of a good user interface, 2014. `http://www.argondesign.com/news/2014/feb/5/5-aspects-good-user-interface/`.

[6] S. Balakrishnama, A. Ganapathiraju, and J. Picone. Linear discriminant analysis for signal processing problems. In *Southeastcon '99. Proceedings. IEEE*, pages 78–81, 1999.

[7] K. S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In *Proceedings of the 7th International Conference on Database Theory*, pages 217–235, 1999.

[8] P. Breuer, C. Eckes, and S. Müller. Hand gesture recognition with a novel ir time-of-flight range camera: A pilot study. In *Proceedings of the 3rd International Conference on Computer Vision/Computer Graphics Collaboration Techniques*, MIRAGE'07, pages 247–260. Springer-Verlag, 2007.

[9] M. Campbell. Apple patents hover-sensing multitouch display, Feb 2016. `http://appleinsider.com/articles/16/02/02/apple-patents-hover-sensing-multitouch-display-`

[10] F.-S. Chen, C.-M. Fu, and C.-L. Huang. Hand gesture recognition using a real-time tracking method and hidden markov models. *Image and Vision Computing*, 21(8):745 – 758, 2003.

[11] CyberGlove Systems. Cyberglove iii. Available at `http://www.cyberglovesystems.com/cyberglove-iii/`.

[12] Fitbit Inc. Fitbit. Available at `https://www.fitbit.com/`.

[13] S. Gupta, D. Morris, S. Patel, and D. Tan. Soundwave: Using the doppler effect to sense gestures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2012.

[14] Z. Halim and G. Abbas. A kinect-based sign language hand gesture recognition system for hearing- and speech-impaired: A pilot study of pakistani sign language. *Assistive Technology*, 27(1):34–43, 2015.

[15] J. Han, L. Shao, D. Xu, and J. Shotton. Enhanced computer vision with microsoft kinect sensor: A review. *IEEE Transactions on Cybernetics*, 43(5):1318–1334, Oct 2013.

[16] R. Harper, T. Rodden, Y. Rogers, and A. Sellen. *Being Human: Human Computer Interaction in 2020*. Microsoft Research Ltd, April 2008.

[17] P.-K. Huang, T.-Y. Lin, H.-T. Lin, C.-H. Wu, C.-C. Hsiao, C.-K. Liao, and P. Lemmens. Real-time stereo matching for 3d hand gesture recognition. In *SoC Design Conference (ISOCC), 2012 International*, pages 29–32, Nov 2012.

[18] R. R. Igorevich, P. Park, J. Choi, and D. Min. Two hand gesture recognition using stereo camera. *International Journal of Computer and Electrical Engineering*, 5(1), Feb 2013.

[19] June Life Inc. June oven. Available at `https://juneoven.com/`.

[20] B. Kellogg, V. Talla, and S. Gollakota. Bringing gesture recognition to all devices. In *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, NSDI'14, pages 303–316. USENIX Association, 2014.

[21] G. D. Kessler, L. F. Hodges, and N. Walker. Evaluation of the cyberglove as a whole-hand input device. *ACM Trans. Comput.-Hum. Interact.*, 2(4):263–283, Dec 1995.

[22] S. A. Khajone, P. D. S. W. Mohod, and P. V. M. Harne. Implementation of a wireless gesture controlled robotic arm. *International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)*, 4(5), May 2015.

[23] K. Khoshelham and S. O. Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2), Feb 2012.

[24] K. K. Kim, K. C. Kwak, and S. Y. Ch. Gesture analysis for human-robot interaction. In *2006 8th International Conference Advanced Communication Technology*, volume 3, pages 4 pp.–1827, Feb 2006.

[25] KinectEDucation. Kinect paint. Available at `http://www.kinecteducation.com/blog/tag/drawing-with-kinect/`.

[26] Leap Motion. Available at `https://www.leapmotion.com/`.

[27] D.-H. Lee and K.-S. Hong. Game interface using hand gesture recognition. In *Computer Sciences and Convergence Information Technology (ICCIT), 2010 5th International Conference on*, pages 1092–1097, Nov 2010.

[28] J. Lee and T. L. Kunii. Model-based analysis of hand posture. *IEEE Computer Graphics and Applications*, 15(5):77–86, Sep 1995.

[29] S. H. Lee, M. K. Sohn, D. J. Kim, B. Kim, and H. Kim. Smart tv interaction system using face and hand gesture recognition. In *2013 IEEE International Conference on Consumer Electronics (ICCE)*, pages 173–174, Jan 2013.

[30] C. Leys, C. Ley, O. Klein, P. Bernard, and L. Licata. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology*, 49:764 – 766, 2013.

[31] T. Li, C. An, Z. Tian, A. T. Campbell, and X. Zhou. Human sensing using visible light communication. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, 2015.

[32] R.-H. Liang and M. Ouhyoung. A real-time continuous gesture recognition system for sign language. In *Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on*, pages 558–567, Apr 1998.

[33] J. Lin, Y. Wu, and T. S. Huang. Modeling the constraints of human hand motion. In *Human Motion, 2000. Proceedings. Workshop on*, pages 121–126, 2000.

[34] T. D. C. Little, P. Dib, K. Shah, N. Barraford, and B. Gallagher. Using led lighting for ubiquitous indoor wireless networking. In *2008 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, pages 373–378, Oct 2008.

[35] R. Lockton. Hand gesture recognition using computer vision. Technical report, Balliol College Oxford University, 2002.

[36] Z. Lu, X. Chen, Q. Li, X. Zhang, and P. Zhou. A hand gesture recognition framework and wearable gesture-based interaction prototype for mobile devices. *IEEE Transactions on Human-Machine Systems*, 44(2):293–299, April 2014.

[37] C. D. Manning, H. Schtze, and P. Raghavan. k nearest neighbor. In *Introduction to Information Retrieval*, chapter 14.3, pages 297–301. Cambridge University Press, July 2008.

[38] R. B. Marimont and M. B. Shapiro. Nearest neighbour searches and the curse of dimensionality. *IMA Journal of Applied Mathematics*, pages 59–70, 1979.

[39] Microsoft Corp. User interface principles. `https://msdn.microsoft.com/en-us/library/windows/desktop/ff728831(v=vs.85).aspx`.

[40] Microsoft Corp. Xbox. Kinect. Available at `http://www.xbox.com/en-US/xbox-one/accessories/kinect-for-xbox-one`.

[41] Nest Labs. Nest thermostat. Available at `https://nest.com/thermostat/meet-nest-thermostat/`.

[42] Nintendo. Wii remote. Available at `https:`

//store.nintendo.com/ng3/browse/productDetailColorSizePicker.jsp?productId=prod150198#.

[43] S. Nirjon, J. Gummeson, D. Gelb, and K.-H. Kim. Typingring: A wearable ring platform for text input. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '15, pages 227–239. ACM, 2015.

[44] Q. Pu, S. Gupta, S. Gollakota, and S. Patel. Whole-home gesture recognition using wireless signals. In *Proceedings of the 19th Annual International Conference on Mobile Computing & Networking*, 2013.

[45] T. Schlomer, B. Poppinga, N. Henze, and S. Boll. Gesture recognition with a wii controller. In *Proceedings of the Second International Conference on Tangible and Embedded Interaction*, pages 11–14. ACM, 2008.

[46] S. Schmid, G. Corbellini, S. Mangold, and T. R. Gross. Led-to-led visible light communication networks. In *Proceedings of the Fourteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '13, pages 1–10. ACM, 2013.

[47] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore. Real-time human pose recognition in parts from single depth images. *Commun. ACM*, 56(1):116–124, Jan 2013.

[48] A. Sridhar and A. Sowmya. *Advances in Visual Computing: 4th International Symposium, ISVC 2008, Las Vegas, NV, USA, December 1-3, 2008. Proceedings, Part I*, chapter Multiple Camera, Multiple Person Tracking with Pointing Gesture Recognition in Immersive Environments, pages 508–519. Springer Berlin Heidelberg, 2008.

[49] T. Starner and A. Pentland. Real-time american sign language recognition from video using hidden markov models. In *Computer Vision, 1995. Proceedings., International Symposium on*, pages 265–270, Nov 1995.

[50] D. J. Sturman, D. Zeltzer, and S. Pieper. Hands-on interaction with virtual environments. In *Proceedings of the 2Nd Annual ACM SIGGRAPH Symposium on User Interface Software and Technology*, UIST '89, pages 19–24. ACM, 1989.

[51] L. Sun, S. Sen, D. Koutsonikolas, and K.-H. Kim. Widraw: Enabling hands-free drawing in the air on commodity wifi devices. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, MobiCom '15, pages 77–89. ACM, 2015.

[52] S. Waldherr, R. Romero, and S. Thrun. A gesture based interface for human-robot interaction. *Auton. Robots*, 9(2):151–173, Sep 2000.

[53] F. Weichert, D. Bachmann, B. Rudak, and D. Fisseler. Analysis of the accuracy and robustness of the leap motion controller. *Sensors*, 13(5), May 2013.

[54] C. Xu, P. H. Pathak, and P. Mohapatra. Finger-writing with smartwatch: A case for finger and hand gesture recognition using smartwatch. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, HotMobile '15, pages 9–14. ACM, 2015.

[55] H.-S. Yoon, J. Soh, Y. J. Bae, and H. S. Yang. Hand gesture recognition using combined features of location, angle and velocity. *Pattern Recognition*, 34(7):1491 – 1501, 2001.

[56] C. Zhang, J. Tabor, J. Zhang, and X. Zhang. Extending mobile interaction through near-field visible light sensing. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, MobiCom '15, pages 345–357. ACM, 2015.

[57] C. Zhao, K.-Y. Chen, M. T. I. Aumi, S. Patel, and M. S. Reynolds. Sideswipe: Detecting in-air gestures around mobile devices using actual gsm signal. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, pages 527–534. ACM, 2014.