Designing a Software Receiver for Gesture Recognition with Ambient Light

by

Dimitar Barantiev
Responsible Professor: Qing Wang
Supervisors: Ran Zhu, Mingkun Yang

Embedded & Networked Systems Group
EEMCS, Delft University of Technology, The Netherlands

A Dissertation
Submitted to EEMCS faculty
Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 19, 2022

Abstract

There is a growing need for touch-free interaction with public utilities such as coffeemakers and vending machines that will help prevent the spread of diseases such as COVID-19. One solution is the integration of embedded gesture recognition systems relying on ambient light. However, existing work so far is found to be inefficient in terms of size, cost and recognisable gestures. This research is part of the development of a smaller and more economical machine learning-powered gesture recognition system using only 3 photodiodes and an Arduino microcontroller. The goal is to design the software for sensor reading, gesture detection and data preprocessing. The resulting receiver samples at 100 Hz, uses an adaptable threshold for identifying gesture endpoints and a mix of FFT, maximum division and Linear Interpolation for signal processing. It is evaluated in two lighting conditions on two distinct gestures and is found to provide a good trade-off between simplicity, real-time processing within milliseconds and robustness against environmental changes. This is achieved with a small RAM memory footprint of only 2 KB and independence of classification backend. The existing design can be further improved in the future through software optimisation and extended environment dynamics support.

1 Introduction

Recent years have seen an increase in the need for touch-free interaction with public utilities such as coffeemakers and vending machines because of their contribution to the spread of many diseases, including COVID-19 [1]. One solution to this problem is the development of embedded gesture recognition systems that can be integrated into those utilities. As a result, people would be able to use mid-air hand movements, reducing the chance of infection.

A class of gesture recognition systems suitable for this purpose are those utilising unmodulated ambient light. They include a set of light detectors which sense the illumination coming from the sun or light bulbs and then track how different gestures' dynamic shadows create a unique change in the sensor readings. Unmodulated ambient light is already in abundance around people every day (see Section 2). Consequently, those systems would require little additional setup or resources after being installed in existing infrastructure and could operate independently, thus providing low cost and ease of integration.

However, existing gesture recognition systems of this kind seem to lack efficiency in terms of hardware cost, size and recognisable gesture set. In order to use ambient light, they require either a solar cell [2], floor-deployed light sensors [3] or a photodiode array, where the number of components ranges from 9 to more than 50 [4, 5, 6]. In all these cases there is a high chance the overall system becomes expensive and infeasible in practice for mass production and integration. In addition, most of those systems' light-sensing and computational hardware is too large. As an exception, one specific system manages to use only a small amount of resources but fails to support more than 4 simple swiping gestures, which does not provide enough flexibility for practical applications [7].

This research is part of the development of a small-scale and cost-efficient embedded ambient light gesture recognition system that uses only 3 photodiodes and a small Arduino microcontroller. Its aim is to design and implement a software-based receiver pipeline that collects real-time gesture data and processes it for classification by a Machine Learning (ML) model. To this end, the research answers the following question:

How to design a receiver for the detection of visible light signals with one Arduino Nano 33 BLE and two or three OPT101 photodiodes, and how to proceed the signals efficiently?

The main challenges associated with this question are as follows:

1. The time it takes to perform gestures varies greatly. Generally, different hand motion speeds are used among users and sometimes even by the same user. As a result, the collected gesture data is of variable length but the target system's ML model requires all input to have a fixed size.

2. Static ambient light intensity in the environment can change dynamically. A room's illumination level can suddenly change, the system can be moved from one building premise to another or the light source can change position. This in turn changes the stable photodiode readings, making the detection of gesture start and end more difficult.

3. Gesture data can be noisy because of moving objects in the background. For example, the motion of other people behind the user can lead to fluctuations in the normal gesture signal. In turn, those may cause ML misclassification if not sufficiently suppressed.

4. Gesture data differs in dynamic range depending on ambient light conditions. Changing the static light intensity also affects the maximum and the minimum read values during a gesture significantly. Consequently, there is more variance between gesture data obtained in different illumination settings, making it harder for the ML model to be trained on it.

5. The receiver pipeline has to provide a fast end-to-end process. In order to be suitable for a practical embedded system, the receiver has to conduct its computations from data collection to final output within milliseconds to avoid stalling the final gesture recognition system.

1.1 Contributions of This Research

The research presented in this paper makes the following contributions:

1. An ambient light receiver design which provides a good trade-off between simplicity, real-time processing, and robustness against environment changes.

2. An ambient light receiver design that is independent of gesture classification backend.

3. A fast and memory-efficient receiver implementation based on that design.

1.2 Research Paper Structure

This paper is structured as follows. Section 2 discusses the technical background of this project. The overall gesture recognition system is laid out in Section 3, together with a further discussion on the research question addressed in the paper. The research methodology and the chosen receiver design are described in Section 4, followed by a brief discussion of specific implementation aspects in section 5. In Section 6 the final receiver is evaluated based on experimental results. Section 7 focuses on the reproducibility aspect of this research. Existing work in the domain of gesture recognition is detailed in section 8. Finally, Section 9 makes conclusive remarks and suggests possible future work related to the ambient light receiver.

2 Background

2.1 Ambient Light

Ambient light is any light in the environment that is not deliberately set up or controlled [8]. This research is based on the use of only ambient light associated with the visible spectrum. This includes, amongst others, sunlight, light bulb illumination, street lamp illumination, etc. The advantage is that all these type of lighting cannot pass through opaque objects, in the case of this project - human hands, and are not so prone to reflections causing multipaths from source to destination as opposed to other types of waves, which makes them a better medium for use in the domain of gesture recognition than radio frequency or sound waves [9].

2.2 Photodiodes

The main project this research is part of utilises photodiodes for gesture recognition. A photodiode is a small semiconductor device which, when operating in its so-called *photoconductive mode* [10], at any point in time outputs a voltage proportional to the amount of light that is cast on it. As a result, when a hand is, for example, swiped above a photodiode, the light falling on it and thus its output first start to decrease, then reach a bottom value at some point when the hand is directly between the light source and the photodiode, and finally increase back to their initial state.

The change in the photodiode's output signal depends also on its sensitivity, which is the ratio between output signal magnitude and amount of incident light. This can be adjusted using a feedback loop [11] with a resistor. The value of that resistor determines how sensitive the photodiode is, and a higher resistance leads to a smaller sensitivity/ratio.

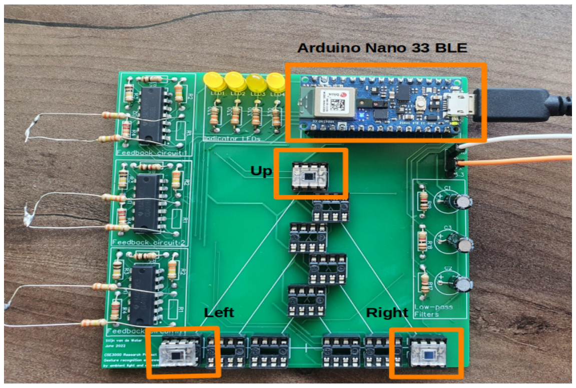

Figure 1: Gesture Recognition Prototype: Arduino Nano 33 BLE (Top) and 3 Photodiodes (Up, Left and Right)

3 System Overview

3.1 System Hardware Setup

As mentioned in Section 1, this research is part of a larger project that tries to implement a small-scale gesture recognition embedded system. The project is done by a team of five people. The target system relies on 3 OPT101 photodiodes for sensing the incident light and an Arduino Nano 33 BLE microcontroller as the computational platform for both the receiver pipeline and the machine learning model. In addition, based on the research work done by another team member, it also makes use of passive elements such as resistors and capacitors, together with a set of transistors to facilitate hardware adjustment of the photodiode sensitivity (see Figure 1).

3.2 System Operation Overview

The final embedded system works in the following way:

1. Continuously reads the photodiode sensors based on a sampling period.
2. Detects when a gesture starts and ends. Collects the photodiode readings between these two time points as the gesture data.
3. Pre-processes the gesture data.
4. Passes the gesture data to a trained machine learning model that determines the input gesture.

3.3 Research Methodology

As part of the overall project, the purpose of this research is to develop the software-based pipeline that reads the photodiodes and handles the edge detection(identifying when the gesture starts and ends), gesture data collection, and signal processing stages of the gesture recognition system. In order for all challenges outlined in Section 1 to be clearly addressed, the research question was divided into the following sub-questions:

1 What sampling frequency should be used for the continuous reading of the photodiodes?

2 How to detect the start and end of a gesture while reading the photodiodes?

3 How to remove noise from the collected gesture data?

4 How to normalise the gesture data?

5 How to convert the gesture data to a fixed length?

6 How to adjust to changes in static light intensity?

Sub-questions 1 and 5 address the first listed challenge. The chosen sampling frequency has to provide a trade-off between data sufficiency for variable gesture speeds(between half a second and two seconds) and lack of gesture data redundancy. Solving sub-question 2 must result in an edge detection algorithm designed for real-time processing. Sub-questions 3 reflects challenge 3 where fluctuations in the data have to be sufficiently reduced. Sub-question 4 is meant to deal with challenge 4, because through normalisation to a fixed value range the receiver output data will be better suited for ML training and classification. Finally, sub-question 6 relates to challenge 2 and solving it will allow the receiver to operate in dynamically changing light conditions.

4 Receiver Design

4.1 Design Methodology

In order to address the sub-questions laid out in the previous section, a literature review of various existing gesture recognition systems' papers was conducted. The sampling frequency, edge detection, noise reduction and normalisation algorithms they use were assessed based on their time performance and memory footprint. This was done in order to determine if they are suitable for the small-scale nature, real-time processing needs and intended use cases of this research target system.

After existing work's documentation was inspected, a few algorithms were considered for the noise removal and normalisation stages. The edge detection algorithm was derived and adapted from that of the GestureLite system [4]. The techniques investigated for noise removal were the Fast Fourier Transform (FFT) [12] and the Discrete Wavelet Transform (DWT) [13] which provide low-pass filtering, as well as the Hampel Identifier [14] that is normally used to detect outliers in time series.

Eventually, only the FFT was incorporated in the final receiver design. This decision was based on its relatively small amount of required calculations for short signals, which meant real-time processing would be easier to achieve. Another benefit of using it was the high availability of already existing libraries providing its efficient computation.

Due to time constraints, the DWT could not be implemented and tested as part of the receiver pipeline, and because of its overall complexity it was discarded as a suitable candidate for noise reduction.

The Hampel Identifier was implemented and tested on real-time collected gesture data. Its performance was found to be of low value because in practice the gesture data did not include visible outliers caused by hardware or environmental noise. In addition, the algorithm was tested to see if it can provide robustness against static light intensity changes during a gesture, but with no success. As a result, the Hampel Identifier was also rejected.

For data normalisation, the considered techniques were the Z-transform [15] and simple division by the maximum of the data series. The former is constituted of two steps - first subtraction of the mean from the data and then division by its standard deviation. However, because photodiode readings were going to be positive integers and the Z-transform could result in negative values, it was deemed more unsuitable for the needs of the system for data variance removal. Consequently, the second approach was chosen. A further discussion on it is provided in Section 4.5.

The final receiver pipeline became a combination of basic techniques, one part based on adapted existing projects' approaches and another based on simple signal-processing algorithms.

4.2 Sampling Frequency

It was established that a sampling frequency of 100 Hz is the best choice for the ambient light receiver. It provides an efficient trade-off between enough data for different gesture speeds and worst-case required memory space. Lower sampling frequencies like 50 Hz can potentially result in gesture data of length less than 20 sampling, which seems to lack enough features for accurate machine learning classification. On the other hand, higher sampling frequencies like the one used by GestureLite can lead to too much data redundancy, resulting in as much as a total of 14000 data points for a single gesture [4], and this wastes much more computational and memory resources than actually needed.

4.3 Edge Detection

The final receiver's edge detection algorithm uses a threshold value per photodiode. It then relies on the fact that when a gesture is being performed, at least one photodiode signal will go under its corresponding threshold, effectively indicating the start point. Similarly, the end point occurs when all photodiode signals return above their thresholds. The data from all photodiodes between those time points is stored as the gesture data.

There were two problems this algorithm had to account for. Firstly, environment noise could cause a sudden drop in one of the photodiodes' signals and lead to a false positive gesture start. Secondly, some gestures like clockwise finger hand rotation were experimentally found to contain intermediate periods of no cast shadow on any photodiode, which could potentially lead to a false positive gesture end.

In order to deal with those issues the algorithm was adjusted to use multiple consecutive samples per photodiode when determining if its signal has gone below or above its corresponding threshold when detecting gesture start and end. That avoided noise in the photodiode data to be classified as a gesture start. Also, the end detection length was chosen to be a few times longer to provide enough time for intermediate zero-shadow periods and prevent misclassified gesture ends. Lastly, to further guarantee that background noise is not seen as an actual gesture, gestures less than 100 milliseconds are rejected by the receiver.

4.4 Threshold Computation

The receiver computes the photodiode thresholds for edge detection in one of two ways. In the first case, it waits until no gesture is performed for one whole second and then uses the median of every individual photodiode time series during that period multiplied by a coefficient between 0 and 1 as the corresponding diode's new threshold. In the second case, if a gesture takes longer than three seconds than it is considered to not be a gesture but a change in static light intensity and then the collected 'gesture data' is used the same way as in the first case to update the thresholds. As a result of this approach, the receiver allows very fast adapting to changes in light conditions with low computational cost.

4.5 Preprocessing

After the gesture data is collected, it is further processed to reduce environment noise, followed by normalisation.

The former stage relies on the use of FFT to remove noisy frequencies. The FFT is an algorithm that maps a time series signal of length N to its frequency domain representation, where the output is also an array of length N but every value reflects how one specific frequency is present in the time series (magnitude and phase). The inverse operation is called inverse FFT (iFFT) and maps back to the time domain.

The preprocessing is comprised of 6 basic steps which require minimum computational time and so provide a high degree of real-time processing support. All steps are performed on each photodiode data series independently:

1. Cut off values in each independent data series which are above the corresponding current photodiode threshold multiplied by a coefficient between 0 and 1, separate from the one used for edge detection.
2. Trim the data series on the right while all photodiodes are equal to the threshold, effectively decreasing the gesture length.
3. Time-stretch the data series up or down to a power of 2.
4. Convert to frequency domain with FFT, keep only low gesture-related frequencies, zero-out irrelevant high frequencies and then convert back with iFFT.
5. Cut off values in each independent data series which are above the corresponding current photodiode threshold.
6. Normalise each data series using its minimum to the range (0,1).

By removing values higher than the gesture threshold in a signal, the receiver leaves only the part of its waveform influenced by the user-cast dynamic shadow. A second cutting off in Step 5 is needed because the FFT introduces small bumps in the flat regions of the original signal that it processes.

The FFT requires its input to have a length that is a power of 2 so step 4 is required and uses Linear Interpolation (see next subsection) to adjust the length accordingly. The FFT stage keeps frequencies no larger than 5 Hz, it being the highest frequency people normally use in hand gestures. As a consequence, the signals are smoothed significantly, reducing data variance and making them better suited for machine learning classification.

The final step scales individual signals based on their minimum so that it is mapped to 0. In essence, each signal is first flipped vertically, then the maximum is found in the resulting data, followed by a division by that maximum, switching the value range from $(0, threshold)$ to $(0, 1)$, and finally the signal is flipped back. The results are two: the peaks of the separate photodiode signals are all mapped to the same value of 0, providing a more environment-independent way gesture data looks; the output value range is suitable for machine learning models.

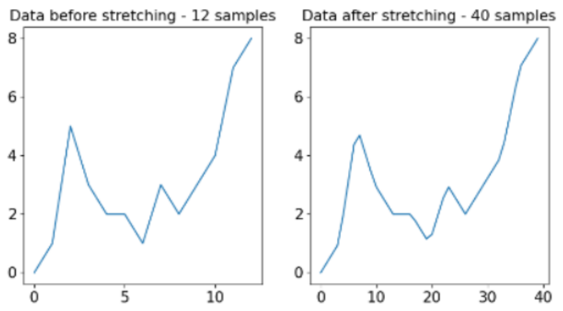

Figure 2: Linear Interpolation technique applied to an example time-series signal. Each output value is computed as the linear interpolation of two values in the input based on the index of the output value.

4.6 Final Receiver Output

As the final receiver stage, the gesture data of all photodiodes is again time-stretched to a pre-established fixed length using Linear Interpolation. Figure 2 provides an example signal transformation based on this approach. Not only is this technique extremely fast and flexible, but it also keeps all important signal information intact provided the output length is large enough, making it a particularly good choice for real-time processing systems like the target of this research.

5 Receiver Implementation

This section provides some additional details in terms of the receiver's software tool basis and parameter tuning for the different stages described above.

5.1 Software Tools

The code base of the receiver was written in C++ combined with the standard framework for Arduino microcontrollers. In addition, three third party Arduino libraries were used: QuickMedian[1], offering a fast, in-place computation of the median of a data series; SimpleTimer[2], providing a basic timer interrupt routine functionality allowing the receiver to achieve sampling time accuracy; and arduinoFFT[3], implementing the FFT and iFFT computations for floating point numbers.

[1]https://www.arduino.cc/reference/en/libraries/quickmedianlib/
[2]https://playground.arduino.cc/Code/SimpleTimer/
[3]https://www.arduino.cc/reference/en/libraries/arduinofft/

5.2 Pipeline Parameters

The edge detection algorithm uses a window of 10 samples to identify a gesture start, which was found to provide enough robustness against background noise fluctuations. For end detection, it uses a window of 50 samples, providing support for gestures of long intermediate lack of shadow. For threshold adjustment, the algorithm takes the median of the collected data as described above and uses 70% of it as the new threshold value, while also using 110% of that threshold for the first cutting-off stage of the preprocessing pipeline. Finally, the current receiver implementation uses an FFT length of 128 and outputs a fixed-width signal of length 100, which is considered to be both space-efficient and not compromising on data sufficiency.

6 Receiver Evaluation

Evaluation Setup: After implementation, the receiver was tested on multiple combinations of gestures and light intensity conditions. In all cases, the setting was indoors, both sunlight and lamp illumination were present as ambient light, and a fixed photodiode sensitivity with a $330k\Omega$ resistor was used. The Arduino microcontroller provided external power with 3.3V to the photodiodes, and their maximum digital output value (the value read by software) was around 800.

Signal Processing Efficiency: Figure 3 shows how the receiver performs for four pairs of gesture and illumination level. It can be easily seen that after processing, fluctuations in the photodiodes signals are suppressed significantly. In addition, the data is converted to a more uniform condition-independent representation with equal scaling in both axes and a common baseline for all photodiode signals.

Time and Memory Efficiency: The end-to-end run of the receiver processing pipeline takes around 20 ms, which in theory satisfies any gesture recognition system's real-time constraints. The software uses small buffers for all its operations based on the assumption that, in practice, all gestures will take at most two seconds. Consequently, its RAM memory

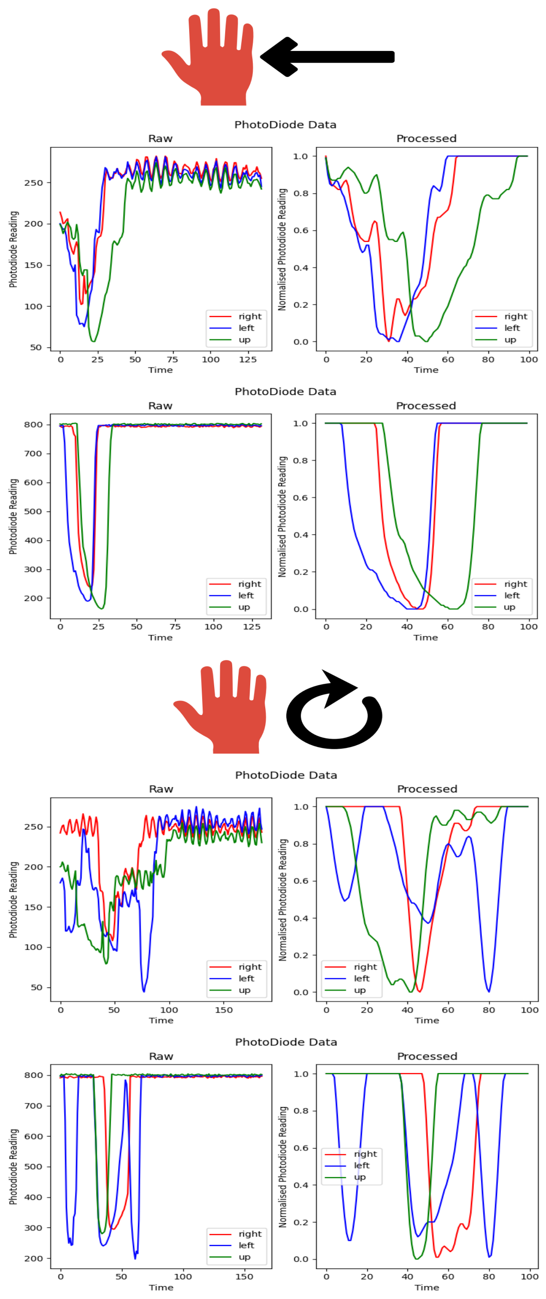

Figure 3: Gesture signals for Left Swipe and Clockwise Hand Rotation passed through the receiver pipeline in 100 Lux (row 1 and 3) and 700 Lux (row 2 and 4). On the left the signal is in the range $(0, threshold)$. On the right the signal is in the range $(0, 1)$.

footprint is around only 2 KB and its machine code takes around 10 KB. Because embedded systems usually have a few hundred kilobytes of both RAM and ROM available in hardware, the receiver software is deemed sufficiently small for usage in this domain.

Optimal Operating Conditions: After testing in the aforementioned setup, the results showed that the receiver is able to operate sufficiently well in the presence of as little as 100 Lux and up to at least 700 Lux. It is expected that in conditions out of this range it can still achieve the same performance but it is required that the photodiode sensitivity be adjusted accordingly to allow gesture shadows to cause enough variation in sensor readings so that threshold-based edge detection is possible.

Disadvantages: The receiver version at the time of writing this paper has the following drawbacks:

1. It does not provide robustness against changes in light conditions during a gesture.
2. It cannot, in general, operate correctly in extremely fast changing lighting conditions.
3. It performs computations even while no actual gesture recognition occurs.

The processing pipeline outlined in the previous section does not account for irregularities like sudden signal drops and rises happening while gesture data is being collected. This can result in gesture output data as the one shown in Figure 4.

In addition, if static light intensity changes at least once every 2 seconds (like in the case of clouds constantly influencing the ground surface shadowing), the receiver does not update its threshold fast enough to allow new gestures to be inputted and efficiently processed.

Lastly, the threshold adjustment procedure takes place at least once every second the overall system is inactive, which wastes some amount of computational resources and power. However, the operation takes only a few tens of microseconds so this is still considered acceptable for the practical use case.

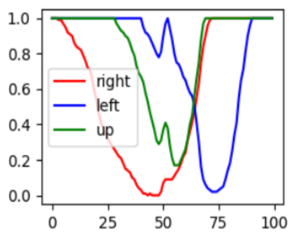

Figure 4: Example of a sudden change in the photodiodes' signals when static light intensity increases right in the middle of a gesture

7 Responsible Research

The results of this research are easily reproducible provided one has the code base[4] and installs the aforementioned Arduino libraries. In addition, one should make sure the light intensity conditions match or are between the ones shown in Section 6.

On the other hand, for the purposes of full disclosure on how the research was conducted, it must be mentioned here that not all possible receiver solutions were tested. In particular, Section 4 already mentioned that the DWT was not implemented due to lack of time. Another algorithm discarded for the same reasons was B-spline interpolation [16], which could provide robustness against variations in light intensity during a gesture but its existence was discovered too late, making it hard for proper evaluation to begin. Finally, some of the existing work-used edge detection approaches were not investigated as part of the receiver once the currently used approach proved to work with reasonable efficiency.

8 Related Work

Many gesture recognition systems using unmodulated ambient light have been developed during the past decade. This section summarises some of them, namely GestureLite [4], LiGest [3], SolarGest [2], ViHand [5], LightDigit [6] and 4-LDR [7].

All systems rely on a large microcontroller like Arduino Due or Uno for computations, and in particular GestureLite even performs the actual preprocessing on a laptop. With respect to light sensing, one of four types of hardware is used - floor-deployed light sensors, which are found in LiGest, a solar cell, used in SolarGest, light dependent resistors, part of 4-LDR, and finally GestureLite, ViHand, and LightDigit make use of a photodiode array.

For edge detection, the solutions either use a dynamically adjustable threshold which is constantly updated by the new sensor readings, as is the case for GestureLite, LiGest, 4-LDR, or a sliding window edge detection, where only the most recent readings determine if a gesture edge is contained in them, which is the approach taken by SolarGest.

In terms of data processing, LiGest and SolarGest use the Discrete Wavelet Transform for noise reduction before edge detection, while ViHand combines a Hampel Identifier for outlier removal with a low-pass Butterworth filter to isolate only frequencies present in a human gesture signal. A common technique for data normalisation among most listed projects is the Z-score transform, which results in data having a normal distribution with zero mean and variance of one. One exception is GestureLite which divides by the maximum of the data.

9 Conclusion

This paper presented a design for a software-based ambient light receiver using photodiodes and running on an Arduino Nano 33 BLE microcontroller that handles gesture detection, data collection and data processing for suppression of environmental influence and normalisation. The design was evaluated in multiple conditions and found to provide a fast and memory efficient end-to-end process with sufficient photodiode signal cleaning performance and condition-agnostic output representation. It can therefore be facilitated by an embedded gesture recognition system designed for real-time processing.

There are two opportunities for improvement of the existing receiver design and implementation by future work. The first one suggests the code can be optimised by switching from FFT/iFFT filtering to convolution with a low-pass filter which is expected to have a polynomially smaller time complexity but requires its coefficients to be adjusted based on the cutoff gesture signal length. The other idea is extending the signal processing pipeline to provide support for additional environment dynamics like light

[4]https://github.com/StijnW66/CSE3000-Gesture-Recognition

changes in the middle of a gesture.

References

[1] EY Knowledge, "In a touchless world, how will you embrace technology?" Tech. Rep., 2020. [Online]. Available: https://www.ey.com/en_gl/innovation/in-a-touchless-world-how-will-you-embrace-technology

[2] D. Ma, G. Lan, M. Hassan, W. Hu, M. B. Upama, A. Uddin, and M. Youssef, "Solargest: Ubiquitous and battery-free gesture recognition using solar cells," in *The 25th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3300061.3300129

[3] R. H. Venkatnarayan and M. Shahzad, "Gesture recognition using ambient light," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 2, no. 1, mar 2018. [Online]. Available: https://doi.org/10.1145/3191772

[4] M.-D. A. Kaholokula, "Reusing ambient light to recognize hand gestures," 2016.

[5] Q. Hu, Z. Yu, Z. Wang, B. Guo, and C. Chen, "Vihand: Gesture recognition with ambient light," in *2019 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation*, 2019, pp. 468–474.

[6] H. Liu, "Lightdigit: Embedded deep learning empowered fingertip air-writing with ambient light," Jan 1970. [Online]. Available: https://repository.tudelft.nl/islandora/object/uuid:afc5471f-c2a3-4ecf-b15e-6edf43a9d22b?collection=education

[7] G. V, S. Salvi, P. Sahoo, M. Dodiya, and S. Gupta, "A shadow based low-cost hand movement recognition system for human computer interaction," in *2021 6th International Conference for Convergence in Technology (I2CT)*, 2021, pp. 1–4.

[8] R. Provost, "What is Ambient Light â Lighting Techniques Explained," 12 2021. [Online]. Available: https://www.studiobinder.com/blog/what-is-ambient-light-definition/

[9] Q. Wang and M. Zuniga, "Passive visible light networks: Taxonomy and opportunities," in *Proceedings of the Workshop on Light Up the IoT*, ser. LIOT '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 42â47. [Online]. Available: https://doi.org/10.1145/3412449.3412551

[10] R. Teja, "What is a Photodiode? Working, V-I Characteristics, Applications," 09 2021. [Online]. Available: https://www.electronicshub.org/photodiode-working-characteristics-applications/

[11] W. Storr, "Negative Feedback and Negative Feedback Systems," 07 2020. [Online]. Available: https://www.electronics-tutorials.ws/systems/negative-feedback.html

[12] Wikipedia contributors, "Fast Fourier transform," 05 2022. [Online]. Available: https://en.wikipedia.org/wiki/Fast_Fourier_transform

[13] ——, "Discrete wavelet transform," 04 2022. [Online]. Available: https://en.wikipedia.org/wiki/Discrete_wavelet_transform

[14] "Filter outliers using Hampel identifier - Simulink - MathWorks Benelux." [Online]. Available: https://nl.mathworks.com/help/dsp/ref/hampelfilter.html

[15] H. Lohninger, "z-Transform." [Online]. Available: http://www.statistics4u.info/fundstat_eng/ee_ztransform.html

[16] Wikipedia contributors, "B-spline," 06 2022. [Online]. Available: https://en.wikipedia.org/wiki/B-spline