Real-Time Gesture Recognition Using 3D Sensory Data and a Light Convolutional Neural Network

Nicholas Diliberti
University of Alabama in Huntsville
Huntsville, AL, USA
nd0006@uah.edu

Chao Peng
Rochester Institute of Technology
Rochester, NY, USA
chao.peng.usa@gmail.com

Christopher Kauffman
University of Alabama in Huntsville
Huntsville, AL, USA
kaufman.christopher.m@gmail.com

Yangzi Dong
Rochester Institute of Technology
Rochester, NY, USA
yd8608@rit.edu

Jeffrey T. Hansberger
Army Research Laboratory
Huntsville, AL, USA
jeffrey.t.hansberger.civ@mail.mil

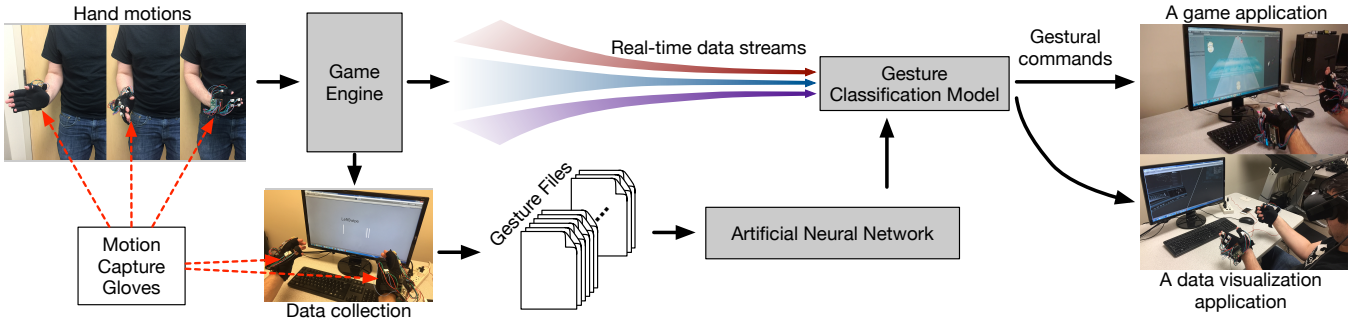

Figure 1: Data pipeline through our system. The system is composed of four major components: motion capture gloves, the game engine, the artificial neural network, and the gesture classification model.

ABSTRACT

In this work, we propose an end-to-end system that provides both hardware and software support for real-time gesture recognition. We apply a convolutional neural network over 3D rotation data of finger joints rather than over vision-based data, in order to extract high-level intentions (features) users are trying to convey. A pair of customized motion capturing gloves are designed with inertial measurement unit (IMU) sensors to obtain gestural datasets for network training and real-time recognition. A network reduction strategy has been developed to appropriately reduce a network's complexity in both depth and width dimensions while maintaining a high recognition accuracy with the classification model produced by the network. The classification model is able to classify new data samples by scanning a real-time stream of joint rotations during the use of the gloves. Our evaluation results expose the relationships between the network reduction hyperparameters and the change of recognition accuracy. Based on the evaluation, we are able to determine an appropriate version of the light network and achieve 98% accuracy.

MM '19, October 21–25, 2019, Nice, France

ACM ISBN 978-1-4503-6889-6/19/10…$15.00
https://doi.org/10.1145/3343031.3350958

CCS CONCEPTS

• **Human-centered computing** → **Gestural input**; • **Computing methodologies** → *Neural networks*; • **Computer systems organization** → Real-time systems; • **Hardware** → Sensor applications and deployments.

KEYWORDS

Gesture recognition; motion tracking; convolutional neural network; human-computer interaction

ACM Reference Format:
Nicholas Diliberti, Chao Peng, Christopher Kauffman, Yangzi Dong, and Jeffrey T. Hansberger. 2019. Real-Time Gesture Recognition Using 3D Sensory Data and a Light Convolutional Neural Network. In *Proceedings of the 27th ACM Int'l Conf. on Multimedia (MM'19), Oct. 21–25, 2019, Nice, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3343031.3350958

1 INTRODUCTION

The use of hand gestures is an important part of natural interactions in everyday body language. Collecting hand motion data and using it to recognize instant gestures that the user performs in real-time data streams are important research challenges for virtual reality, human computer interactions, human behavior studies, and animations. In the realm of hand gesture recognition, there are

two archetypes – static, where hands are at specific postures, and dynamic, where hands perform specific actions and move quickly through a series of positions. While static hand gestures are relatively simple to detect using the angles between different parts of the user's hand, dynamic gestures are not so simple to detect. Different users can start, perform, and finish a gesture in a variety of different angles and speeds, resulting in an explosion of possibilities that is infeasible to handle by enumeration.

Vision-based systems such as Kinect and Leap Motions have been used to detect and recognize gestures, but they suffer the limitations of self-occlusion and a limited range of space in which the user can perform. Flex sensors have been used in sensor-based motion tracking systems such as CyberGlove [43]. A flex sensor measures bending or flexing based on varied resistance. They are known to be costly and fragile, and they tend to produce nonlinearity or errors between sensor flexion and resistance variation after a long-time use due to the change of the sensor's flexibility [8, 33, 34] . In this work, we build a custom data glove using inertial measurement units (IMUs) to capture both static and dynamic hand gestures.

To recognize hand gestures for real-time applications, we propose a light neural network that trains on and then classifies hand motions captured from our custom glove into appropriate gesture categories. While popular neural networks have shown great accuracy in classification, the immense amount of computation required for a single pass through the network makes them unsuitable for real-time applications. Nevertheless, we wished for versions of these networks that could compress the proven effectiveness of their architectures into the memory and time constraints of a real-time desktop application. Our contributions are two-fold:

(1) We built a low-cost data glove to capture rotation angles of the palm and fingers in real-time. It wirelessly connects to the system and produces motion data for gesture recognition. We used a two-step calibration method to precisely match the initial posture of the virtual hands to the user's physical hands.
(2) We developed a light convolutional neural network as a classification model to detect hand gestures in real-time. We recorded 6,700 examples of different users performing these gestures for the network training. The network is able to find the archetypal features of each gesture and classify new data samples by scanning a real-time stream of joint rotations.

2 RELATED WORK

Related to our work, this section reviews several existing approaches for motion capture and gesture recognition, involving both hardware and software development.

2.1 Motion Capture

Researchers have proposed handheld and wearable motion capturing devices that are suitable to interact with virtual and augmented environments, each carrying limitations. Commercial devices used in research of gesture recognition include Wii controller [35], Leap Motion [7, 24, 28], Kinect [27, 40], and CyberGlove [18]. A Wii controller would not capture fine-scale motions on fingers. A CyberGlove is costly and may give a low accuracy when measuring large curvatures on fingers. The Leap Motion and Kinect are optical-based devices, so their tracking accuracy may vary based on lighting conditions and the capturing range of cameras.

An early work by Keir et al. [17] presented a low-cost handheld device for users to perform wand-like motions, but it did not detect or recognize finger-level movements. Jeong et al. [15] used film materials with carbon particles to build a glove for finger-gesture recognition, but their glove had problems of capturing speed and stability, so it was not suitable for real-time interactions. Wang and Neff [41] presented a data-driven calibration approach applicable to general hand capture hardware that mapped raw sensor data to the rotations of hand's joints. Georgi et al. [10] developed a wearable system that captures bone movements using IMUs and muscle activities using electromyography. They directly applied a Hidden Markov Model over 3D signals to build the gesture classification model rather than first articulating a hand representation. Their approach did not consider correlations among signals, which should be intrinsically consistent to the bone relations in the upper-body skeletal hierarchy. Groves [11] presented a detailed reference about using IMUs. Shen et al. [36] used a soft bending, low-cost sensor to build a hand capturing glove, but there was not any evidence about the glove's capturing speed. Calella et al. [4] used raw data from IMUs to recognize pointing-type gestures using a thresholding method. Their approach did not capture fine-scale motions on fingers and did not show potentials of capturing a wide range of gesture types.

2.2 Gesture Recognition

Machine learning and network-based algorithms have been commonly used for gesture recognition. Alavi et al. [1] evaluated support vector machines and artificial neural networks for gesture recognition using the data collected from wireless motion sensors. Xu et al. [42] trained a neural network with one hidden layer to recognize hand gestures and applied it into a virtual reality environment, but their method fails to support interactive requirement due to the slow recognition process. Song et al. [38] developed a continuous gesture recognition system that combined 3D body pose estimations and hand pose classifications. Neto et al. [30] proposed a continuous spotting solution using two artificial neural networks to recognize communicative and non-communicative gestures in real-time for robot controls. Rosalina et al. [32] proposed an artificial neural network trained with colored glove images captured by web cameras, which can recognize static alphabetical and numerical signs. Luzhnica et al. [25] used a customized glove and extracted the gesture features from real-time data stream using a sliding window, but the requirement to fill the sliding window with a sequence of data frames caused a recognition delay. Ma et al. [26] collected muscular activity data and used the data with a trained convolutional neural network to classify gesture signals. However, the lack of training samples and explicit muscular activity errors caused issues in the classification.

3 SYSTEM OVERVIEW

Our system is composed of four principal components: the custom gloves (for wireless hand motion capturing), the game engine (for graphics and hand articulation), the artificial neural network (for

learning gestural features from collected training sets), and the classification model (for real-time gesture recognition). The data flow through our system is illustrated in Figure 1. Individual sensors in the glove calculate and report orientations of hand joints. These orientations are sent to the game engine, which performs inverse kinematics to create a three-dimensional hand model which matches the sensor information. To obtain a classification model for gesture recognition, a data collection session is performed to store gesture data in files. These files are used to train the artificial neural network. The occurrence of a gesture can be identified by the classification model, and this gesture is then used to command the application. The communication between the glove and the game engine is established with a Bluetooth connection. The communication between the game engine and the classification model is over local sockets.

4 CUSTOM GLOVE AND VIRTUAL HAND ARTICULATION

This section describes the design of the custom glove with IMU motion sensors and an articulation method that maps the sensors' orientations onto a rigged hierarchical structure of hand joints.

4.1 Custom Glove

We designed the glove to be lightweight, low-cost, and wearable. The glove captures and sends hand motions wirelessly and in real-time. We ahered IMU sensors on the fabric surface of the glove to track rotations of the palm and rotations of the joints on the thumb, index finger, and middle finger. These three fingers have been proved having a high degree of finger independence [21] and the most expressive movement capability [14]. Raw data returned by IMUs is processed by the fusion algorithm on the microcontroller and can be converted into rotation values in the format of Euler angle or quaternion. In our work, we chose to use quaternions to represent the rotation values because quaternions avoid the potential gimbal lock issue and produce singularity-free rotating trajectories, which we will otherwise encounter with Euler angles.

Figure 2 shows a picture of the prototyped glove and the design diagram. The main hardware components of the glove are a series of IMU sensors, a microcontroller, an I^2C multiplexer, and a power supply. In the glove, rotations captured by the IMUs are sent to the host PC through an established Bluetooth communication.

We chose the Bosch BNO055 intelligent 9-axis sensor [9]. We used a total of seven sensors: one sensor is placed on each proximal phalanges of the thumb, index finger, and middle finger; one sensor is placed on each intermediate phalanges of the index finger and middle finger; one sensor is place on the distal phalanges of the thumb; and one sensor, combined with an onboard microcontroller, is placed on the back of the hand to capture the rotation of the wrist. A custom Printed Circuit Board (PCB) was designed and manufactured in order to obtain the smallest usable platform upon which to run a single BNO055 sensor. The miniaturized PCB is necessary so that the glove is not cumbersome to wear with seven sensors attached. Of particular note is the removal of the external timing crystal, additional voltage regulators, and isolation circuits used by the original Adafruit BNO055 PCB. The custom PCB is capable of returning data requested by the microcontroller through a standard I^2C port. An 8-Channel I^2C multiplexer is used to expand the communication with sensors. In our design, we chose to use the Bluetooth wireless connection method because of its potential to reach a lower power consumption level than Wi-Fi.

4.2 Virtual Hand Articulation and Calibration

The rotation values captured by the glove control the joints of the virtual hand in a simulated 3D environment. A 3D anthropometric polygonal mesh is used as the shape template of the virtual hand to determine constant parameters such as hand size and bone length; thus, there is no need to perform a runtime estimation of these parameters. The joints of the virtual hand are articulated in a kinematic chain so that hand postures and motions are associated to relative bone rotations in the hand skeleton.

The IMUs' quaternions are in an Earth-relative reference frame. They need to be converted with respect to a local reference frame, so that the rotation axes of these quaternions are referenced to the direction that the user's hand is postured towards the display monitors or the initial direction of the camera. To achieve this, the wrist is treated as the root joint, and each IMU's quaternion is referenced to the wrist using Equation 1, where Q_i^{IMU} is the quaternion of a bone associated with an IMU, and Q_w^{IMU} is the quaternion at the wrist.

$$Q_i^{IMU'} = Inverse(Q_w^{IMU}) \times Q_i^{IMU} \tag{1}$$

A potential tracking accuracy issue may arise because the IMU's remapped device axes may not perfectly align to the bone's local rotation axes (relative to its parent bone). This is usually caused by displacement of IMUs on the fingers. To solve this issue, a calibration session is added to match the hand posture to the the posture of the anthropometric hand mesh. During the calibration, a rotation offset is calculated between the IMU's device axes and the bone's local axes using Equation 2, where $\widetilde{Q_i^{anth.}}$ represents a bone's local quaternion from the anthropometric hand mesh (the template hand), and $\Delta\widetilde{Q_i}$ represents the rotation offset in the bone's local domain. During the runtime, a bone of the virtual hand corresponding to an IMU-attached bone will be driven by the local quaternion of $\widetilde{Q_i^{bone}}$ using Equation 3.

$$Calibration: \Delta\widetilde{Q_i} = Localize(Q_i^{IMU'}) \times Inverse(\widetilde{Q_i^{anth.}}) \tag{2}$$

$$Runtime: \widetilde{Q_i^{bone}} = Inverse(\Delta\widetilde{Q_i}) \times Localize(Q_i^{IMU'}) \tag{3}$$

5 DATA CONFIGURATION AND RECOGNITION

5.1 Gesture Data Format

A hand gesture consists of a sequence of continuous frames, with each frame of data consisting of transformation attributes of both left and right hands (16 floating-point numbers per hand). Since the transformation attributes of the two hands are identical, here we describe those attributes for one hand in detail. Figure 3 illustrates the transformation attributes we recorded from the right hand. Three values are used to represent the displacement of the hand in 3D space, represented as an (x, y, z) distance from the hand's

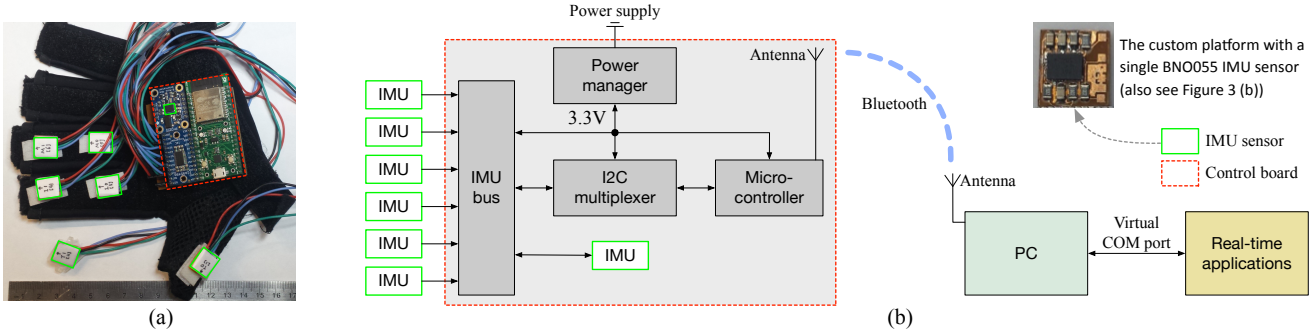

Figure 2: The prototyped glove and the diagram of the hand motion tracking system. (a) shows the picture of the prototyped glove with a total of seven IMU sensors (six standalone sensors and one integrated sensor with the control board). (b) is the diagram shows the connection of the sensors through the IMU bus and I^2C mutiplexer to the microcontroller. The glove is wirelessly connected to the PC and real-time applications through Bluetooth.

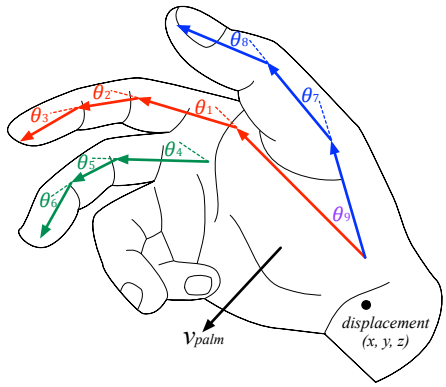

Figure 3: The illustration of transformation attributes of the right hand saved in the gestural data file. The dotted lines correspond to the default postures of finger knuckles at time of calibration.

location at the time of calibration. Four values of a quaternion are used to represent a vector coming out of the palm, denoted as v_{palm}. Quaternions are used to represent the orientations of three knuckles of the index finger (θ_1, θ_2, and θ_3), the orientations of three knuckles of the middle finger (θ_4, θ_5, and θ_6), the orientations of two knuckles of the thumb (θ_7 and θ_8), and the angle between the index finger and the thumb (θ_9), respectively.

We format each gesture recording a data file. Each row in the file is a frame of data that represents the current postures of two hands, as shown in Equation 4. We collect a frame every 0.02 seconds. Our neural network uses a 50-frame chunk for each gestural instance, representing a timespan of 1 second.

$$\underbrace{\{displacement, v_{palm}, \{\theta_1, \ldots, \theta_9\}}_{\text{left hand}}, \underbrace{displacement, v_{palm}, \{\theta_1, \ldots, \theta_9\}\}}_{\text{right hand}} \quad (4)$$

We also appended each frame with the value changes from the previous frame. Smaller changes will reflect a higher similarity in hand posture for the duration of a gesture. By doing this, it is easier for the network to differentiate between static and dynamic gestures. This also provides the network with data specifically indicative of motion, allowing for easy differentiation between static and dynamic behavior.

5.2 Gesture Vocabulary

Interaction-based gestures are composed of both static and dynamic gestures that are meaningful actions and can be used for commanding. To our knowledge, the literature does not have a standard format for defining such gestures for human-computer interactions. To define a gesture vocabulary for interaction, we borrowed the gesture design concept for sign language [23] and applied the literature about gestures in speech [2, 5]. In this work, hand gestures are *semaphoric gestures* that convey a standardized intent when performed [12]. The vocabulary is centered around the use of a neutral supported position, which prevents long-term usage from fatiguing the user's arms. [13].

The following one-handed gestures are used in our system:

- Come – The user holds their hand out with their palm facing upwards, then curls their fingers towards their palm.
- Go – The user holds their fist out with their palm facing downwards, then uncurls their fingers until horizontal.
- Stop – The user holds their hand out with their palm facing forward.
- Swipe – The user holds their hand out with their palm facing towards the centerline of the body, then pushes their hand from outside the body towards the inside of the body.
- Point – The user holds their fist out with their index finger uncurled until straight.
- ThumbsUp – The user holds their fist out with their thumb on the topside of their fist, then extends the thumb until straight.
- Pinch – The user touches the tips of their thumb and index finger together, and extends their other three fingers.
- Twist – The user curls their fingers into the pailm sans their thumb and index finger, which grasp an imaginary knob. The palm begins facing downwards, is rotated upwards.

Additionally, the following two-handed gestures are used:

- ThisWide – The user holds their palms towards each other with their extended fingers pointing upwards.
- ShoveRight – The user starts with both hands facing palm-forward, then rotates both hands to the right and bends them forward at the wrist.
- ShoveLeft – The user starts with both hands facing palm-forward, ten rotates both hands to the left and bends them forward at the wrist.
- Clap – The user hits the palms of their hands together.
- ThisTall – The user holds both of their hands horizontally with the palms facing down, then turns their wrists until each hand's fingers point towards the center line of the body.
- SnapInHalf – The user brings their clenched fists together, then pivots them downwards and away from one another.
- Nothing – The user remains idle in any position that does not match some other static gesture.

5.3 Data Collection and Augmentation

We implemented an interface tool that integrates the glove and the Unity engine. The tool prompts the participant for a specific gesture that he or she needs to perform, and then records the participant's hand motion. The tool requires the participant to start in a neutral hand posture prior to perform a gesture, where the elbow is tucked into the body and the hands are suspended with the palms facing inwards and keeping a short distance in-between. The tool's prompt during the recording includes three components: (1) text on the screen indicating the type of gesture to perform, (2) a vertical line moving from left to right across the screen, which indicates the time period for the participant to perform any movement, and (3) a highlighted region in the middle of the screen, specifying a the time frame into which the user ought to confine their movement. The total time for the vertical line to move across the screen is set to 4 seconds. The highlighted region in the center of the screen occupies 25% of both the total screen space and total time.

It is important to increase data diversity, as the gestural features of interest need to be recognized from both different speeds and magnitudes of motion. To increase data diversity and prevent the network from overfitting, we manufacture new data using the existing data we have collected. This is a known methodology named *data augmentation* [6, 31]. First, we isolated all gestures that use only one hand (Come, Swipe, etc.) and split the recording into active hand and non-active hand data. Then, all elements of active hand data are paired exhaustively with the elements of the non-active hand data, which creates new recordings that contain both the original gesture and all manner of different resting positions for the non-active hand. Second, the frames in the recordings are shifted by some integer in [-20, 20], representing a shift of up to 0.4 seconds. Third, the value for each component of the frame is perturbed by some percent in [-20%, 20%] of the standard deviation for that component across all frames in the recording. These data augmentation steps are used turn one recording into many network inputs, while preserving the gesture each input ought to be recognized as.

5.4 Thresholding System

In order to detect gestures from real-time data streams, we take a one-second history of the user's hand readings and present it to the classification model produced by the neural network. The classification model then returns gestures confidences that have been through the *Softmax* function, which puts each confidence on a scale from 0 to 1 and makes their summation equal 1.

As the gesture vocabulary grows, several gestures may look very similar, resulting in false positives when trying to recognize them using the neural network. To stop the system from signalling false positives, we combined finite-state machine logic with a confidence thresholding system. In this system, the user starts in a "Nothing" state, symbolizing that they are not currently performing any gesture. As soon as a gesture's confidence level rises above its confidence threshold, the system changes its state to that gesture. Once the system's state is a gesture, one of the following behaviors will activate. If the gesture is non-repeatable / instantaneous, no additional gestures can be detected until the system first returns to the "Nothing" state. If the gesture is repeatable / sustained, then on every frame where the gesture still lies above its threshold, it is signalled again. Like with non-repeatable gestures, no other gestures can be detected after a repeatable gesture until the system returns to the "Nothing" state.

We derive these threshold values using a percentile of the true-positive readings for each gesture. For each gesture, we load all the files where the neural network correctly identifies it. Then, we measure the confidence value by which the network arrived at its decision. Among these readings, we sort the results and take the percentile as the threshold value for each gesture.

5.5 Gesture Forecasting

By leveraging the mathematical properties of physical movement, we are able to reduce our gesture recognition latency by forecasting transformation values in the hand model. Unlike pixel values in a picture, which are non-differentiable and discontinuous, the data in our recordings represents angles and orientations, which are both differentiable and continuous. By taking a derivative of the end of the data stream, assuming it to be constant, and extrapolating frames into the future, we can forecast the values and use the augmented data stream to detect gestures more quickly.

When forecasting, we first decide on whether we are using a first or second derivative. The first derivative would correspond to velocity, which holds steady during the middle of a gesture, while the second derivative would correspond to acceleration, which holds steady during the beginning and end of a gesture. Next, we set a hyperparameter that indicates the number of frames which we will be forecasting. Frame by frame, we forecast future readings using the computed derivative and the end of the current data stream. Finally, we cut the oldest frames from the data stream to fit the neural network's input size, then feed the forecast-augmented frame data into the neural network.

6 LIGHT CONVOLUTIONAL NEURAL NETWORK

We came to understand that humans do not recognize or perform gestures based on a stringent set of rules [16]; instead, both the actor and the viewer translate the general meaning of the gesture into a hand motion that varies in position, orientation, speed, and fluidity [16]. In order to achieve maximum system accuracy, we

needed to employ a strategy that could take the physical motion of the user's hand and derive the high-level intention they are trying to convey. We chose neural networks for this task, as they are able to form models that derive high-level relationships from low-level data, then classify inputs based on which relationships are present.

The design of our neural network needed to reach two goals: (1) The classification model produced by the network needs to achieve high accuracy for gesture recognition, and (2) The classification model also needs to run at a high speed for real-time applications, so that it will not impair the user's operation of the system.

For our network architecture, we chose to use a convolutional neural network (CNN), which have been applied to perform visual-type tasks. There are similarities between datasets for visual-type tasks and the dataset used in our gesture recognition task. The layers of operation in our CNN is described in Section 6.1. The core of our CNN is the convolutional operation on the rotation values of the joints, which is computationally expensive. Consequently, it would prevent the trained classification model from the use in real-time applications. Section 6.2 presents our method to reduce the complexity of the CNN while preserving the accuracy.

6.1 Layers of Operation in CNN

The design of our CNN follows the methodology presented in AlexNet [20]. Training the CNN with the gestural dataset is an iterative process. At each iteration, recording data points are transformed by convolutional filters across the entire neural network, and then values of parameters are updated with gradient descent and used at next iteration. There are several layers of operation, each of which takes in the output from the previous layer and performs a specific computation. Our network has two "learning" layer types – convolution and dense, and one "computation" layer type – max-pooling.

Convolution is an operation whereby a subject is tested for similarities against a set of filters. The operation is performed by "sliding" the filters over each possible position in the subject, and at each position measuring the similarity between the captured region and the filter. Each convolutional layer in the network has a limited number of filters, and is responsible for taking the output from the previous layer, performing convolution on it, and then signalling the presence or absence of the features represented by its filters to the next layer.

The dense layer takes the output from the previous layer and feeds it into a number of neurons, which are independent units that each represents a weighted sum of the input values. Then, the layer passes each of these weighted sums through the ReLU non-linear activation function [29]. Dense layers are placed near the end of our network due to their lack of specialized features and high capability to discover high-level relationships among outputs from the previous layer.

Max-pooling is an operation to reduce the size of the data frame travelling forward in the neural network, which forces it to ignore low-level differences among samples and discern high-level meaning from the general form of the data. This drives the network away from rote memorization of coincidental data patterns and towards a more human-like understanding of how low-level features work together to communicate a gesture.

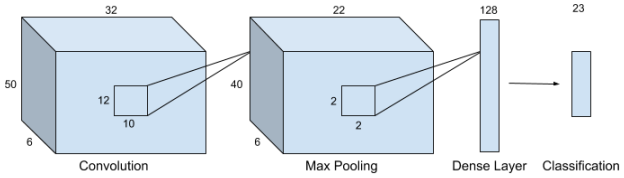

Figure 4: The minimized CNN used in our system for gesture recognition. In comparison with the AlexNet, the light CNN has reduction ratios with $DP = 20\%$ and $RF = 4$.

The outputs of the neural network are fed through a softmax function [3] to transform them into confidence values. These confidence values fall in the range 0 to 1 and sum to 1, giving us a probability distribution of what gesture the user is performing.

6.2 Network Reduction

We reduce the network's depth and width to make them suitable for being integrated with real-time gestural-based user interfaces, all while maintaining the accuracy and essential characteristics of the architecture that provide its unique advantages. The *depth* of the network refers to the number of layers between the input neurons and output neurons The *width* of the network refers to the number of filters employed in each layer that affects the number of neurons produced after the layer.

Decreasing either depth or width values within the architecture of a network increases speed, and this increase is proportional to the number of neurons that have been removed. We kept track of our reductions to the network using two hyperparameters. The first is the *depth percentage* (DP), which is a percentage between 0% and 100% representing the percentage of layers that remain in the network after reduction. The second is the *reduction factor* (RF), which represents the width-reduction magnitude expressed as $neuronNum_{new} = 2^{-RF} neuronNum_{old}$. In our procedure, we first minimize the depth of a network, then minimize its width. Figure 4 shows the light CNN used in our system to train the classification model for gesture recognition.

6.2.1 Depth Reduction. For each network, we first adjusted the aspect ratio of both the input data frame and the convolutional windows to match the aspect ratio of our input data. The total number of input data elements and neurons on each layer was preserved. We remove all layers that are of a higher resolution than our input data, then add back as few layers as possible until we clear a minimum accuracy threshold. These layers are added back from the end of the network towards the beginning, in order of ascending resolution. Conceptually, removing depth from the network reduces the degree to which it can understand high-order relationships amoung the input values, and thus there theoretically exists a minimum depth needed to understand the complexity of the data.

The amount of depth remaining in the network is governed by the DP hyperparameter. An important note is that in some architectures, multiple layers work together to form a single operational unit. For example, GoogLeNet [39] is constructed not of individual layers, but of nine-layer modules called Inception modules. In reducing the depth of the networks with sub-components, we did not

break up these operational units, instead treating them as a single layer to be added or removed.

6.2.2 Width Reduction. We remove neurons from each layer of the neural network using a scheme appropriate to that layer's architecture. For convolutional layers, we reduce the number of filters, and for dense layers we reduce the number of neurons. Conceptually, removing width from the network reduces the number of features it can recognize, and thus there thereotically exists a minimum width needed to form a sufficient set of baseline features.

We reduce the width of the network by incrementing the previously-defined RF, each point of which represents a halving of the number of neurons in the layer. We reduce the number of filters instead of the dimension of each filter because (1) the filters represent an import facet of the network architecture – the size of the features being extracted, and (2) we relied on the dimensionality reduction caused by the convolutional layers in order to arrive at our minimum depth. Changing the size of the convolutional filters would alter those sizes, meaning we would have to start the algorithm again with a different set of resolutions.

6.2.3 Detail of Reduction Algorithm. To begin network reduction, we first determine a minimum required accuracy and a maximum allowable time for the computation of a single frame of input data. For our networks, we chose 98% percent as our minimum accuracy and one-sixtieth of a second (a standard montior's refresh rate) as our maximum allowable time. Second, start with minimal depth and maximal width for the network. Third, we train a network using the current DP and RF. If it satisfies our acceptance criteria, we continue to the next step. If it does not, we add one layer at a time to the network until either the acceptance criteria are met or there are no additional layers left to add, in which case the architecture is deemed inadmissible and the algorithm terminates. Fourth, we take this minimal-depth network and begin increasing its RF. We train the network after each increase to ensure that it still passes the acceptance criteria. The final RF is decided by either failing the acceptance criteria or by reaching a network's maximum allowable RF, determined by the smallest neuron count from among its layers. For example, a network with a convolutional layer of 64 filters could not accept a RF greater than 6, since each convolutional layer requires at least one filter. Finally, this network with minimal DF and maximal RF is saved to the disk.

7 EXPERIMENT AND RESULTS

We used the custom sensor gloves to record hand motions and saved them into files in the format described in Section 5.1. A total of 15 volunteers participated in the data collection process. We collected 20 recordings for each of the 23 gestures in Section 5.2 from each participant. Gestures were prompted in a random order to prevent volunteers from pre-emptively positioning for the next gesture to be recorded. We obtained 6700 recordings and partitioned them into 5360 training files and 1340 testing files, a 4 : 1 split. The networks were trained for a total of 15,000 steps. We used the technique of early-stopping, where the network was saved every time it hit a new highest accuracy, and then the final saved network is taken as the training output. Our training routine ran with a learning rate of 0.0001 and used the Adam optimizer [19].

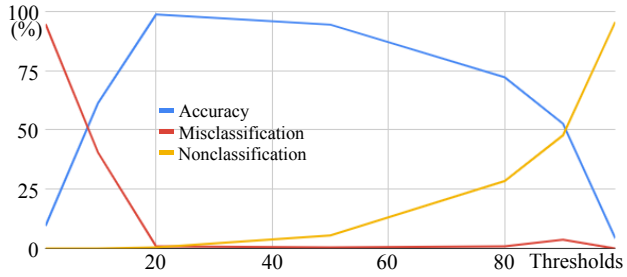

Figure 5: Results showing accuracy, misclassification and nonclassification over various thresholding levels.

7.1 Evaluation on Network Reduction

Our reduction algorithm increased our network's speed. Table 1 shows that even amoung high RFs, it is possible to halve the parameter count and double the execution speed on the AlexNet architecture using DP=100% and RF=6 instead of DP=20% and RF=4. As RF holds constant and DP increases, the number of adjustable parameters in the network decreases. This is because in a reduced-width network, the parameter count is dominated by the first dense layers, where the output of the convolutions is flattened before entering. When using very few filters in the convolutional layers, the result is that the loss in resolution outpaces the gain in filters, reducing overall parameter count.

Our final architecture, AlexNet with DP=20% and RF=4, exhibited a 98% accuracy on the testing data and a softmax cross-entropy reading of 0.1713, achieved after 9,390 steps of training.

7.2 Evaluation of Thresholding System

To choose the thresholds for the state machine detailed in Section 5.4, we used the true-positive rates of the recordings where a gesture was successfully identified. When thresholds were set inappropriately, two types of errors arose. The first was misclassification errors, where a gesture would be recognized as happening, but misidentified. Misclassification errors occurred when either noise generated during a "Nothing" pose or false-positive readings. The second was nonclassification errors, where no gesture was recognized as happening. Nonclassification errors occurred when a user could not clear the threshold of a gesture while performing it.

Figure 5 shows that, as expected, misclassification errors dominated when thresholds were too lenient, and nonclassification errors dominated when thresholds were too strict. Using the AlexNet (DP=20%, RF=4) network, we determined the optimal threshold percentile was 20%.

7.3 Evaluation of Gesture Forecasting

We found the optimal value for the number of forecasted frames by searching linearly through the possible values of 1 (0.02 seconds) to 50 (1 second). The results are shown in Figure 6. Using the first derivative to extrapolate frames is more accurate than using the second derivative. As the number of forecasted frames increases under the first derivative, the accuracy of static gestures rises while the accuracy of dynamic gestures falls. This is because extrapolating dynamic movements will cause them to become exaggerated

DP	RF	Accuracy (%) | Speed (ms) | # of Parameters (million)			
		AlexNet [20]	LeNet [22]	VggNet16 [37]	GoogLeNet [39]
20%	6	92.01 | 4.18 | 6.23	81.32 | 3.91 | 0.71	88.58 | 6.76 | 22.74	70.19 | 6.91 | 1.05
60%	6	95.19 | 5.69 | 6.01	77.32 | 4.54 | 0.66	60.86 | 9.10 | 19.83	11.77 | 9.19 | 0.06
100%	6	80.22 | 7.06 | 5.16	77.32 | 4.54 | 0.66	49.44 | 12.05 | 16.63	11.77 | 9.19 | 0.06
20%	4	98.03 | 8.94 | 98.73	81.32 | 3.91 | 0.71	88.58 | 6.76 | 22.74	87.62 | 7.91 | 17.41
60%	4	97.50 | 9.48 | 96.12	77.32 | 4.54 | 0.66	60.86 | 9.10 | 19.83	63.77 | 9.27 | 0.97
100%	4	89.80 | 10.71 | 82.58	77.32 | 4.54 | 0.66	49.44 | 12.05 | 16.63	63.77 | 9.27 | 0.97

Table 1: The accuracy, speed, and trainable parameter count of four networks under different combination of DP and RF values.

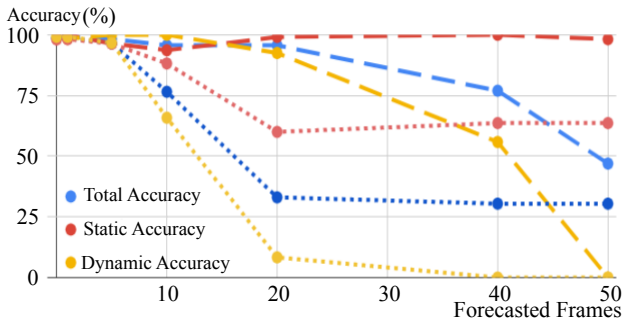

Figure 6: Results of various forecasting thresholds using the first derivative (dashes) and second derivative (dots).

and to pass the stopping point of a normal human hand, while extrapolating static motions will cause the network input to be filled with multiple extremely-similar frames.

Forecasting out to 5 frames using the first derivative retains the accuracy of the original network while allowing gesture recognition to take place 0.1 seconds more quickly. Going farther incurrs a small accuracy penalty out to 20 frames, which retains a 95.60% recognition accuracy while forwarding recognition by 0.4 seconds. Beyond 20 frames, the accuracy drop-off is severe with regards to dynamic gestures for the system to remain viable, leading all the way down to 0% dynamic recognition at 50 frames. Thus, we use the smaller forecasting value of 5 frames. If we were able to increase the network's reliability or be given leeway in terms of accuracy, we could choose the larger value of 20 frames.

7.4 Overall Performance and Comparison

Based on our analysis, we decided on values of the 5 hyperparameters: DP, RF, threshold percentile, forecast frames, and derivative, which are respectively equal to 20%, 4, 20%, 5, and first derivative. We achieved an accuracy of 98% and a speed of 8.94 milliseconds per frame, with 0.1 seconds of forecasting.

The results of our method are compared with the results of other methods in Table 2. Our method out-performs each other method in at least one area – the required number of input sensors, the size of the vocabulary, or the recognition accuracy. The most notable comparison is to the work of Luzhina et al. [25], which has both a larger vocabulary and recognition accuracy, but requires nearly double as many sensors to achieve these results.

Procedure	# of sensors	# of gesture types	Accuracy (%)
Our method	14	23	98.03
Alavi et al. [1]	5	12	100.00
Neto et al. [30]	24	10	99.80
Luzhnica et al. [25]	26	31	98.52
Xu et al. [42]	18	15	97.96
Song et al. [38]	15	24	86.35
Ma et al. [26]	4	8	81.52
Rosalina et al. [32]	1 (camera)	40	88.89

Table 2: Results of our method compared to other methods.

8 CONCLUSION AND FUTURE WORK

In this work, we have created a system for real-time gesture recognition via neural network, Based on the results and our analysis, we can draw several useful conclusions. Our network-reduction algorithm is capable of providing marked increases in speed while still keeping the same accuracy of larger, slower networks. Tuning the five hyperparameters of the network allows users to trade among execution speed, recognition accuracy, and forecasting distance as they see fit. Our finite-state-machine logic functions well for eliminating misclassification and nonclassification errors.

In the future, we plan to expand the pool of training data and include a more diverse cast of users, so that different body types and gesturing styles can be represented. We will test our reduction algorithm on more complex networks at their full size on capable hardware. The forecasting system can be improved by setting limits on the predicted values such that no impossible hand positions can be forecasted. Additionally, the accuracy of the second-derivative forecasting can be improved by implementing dropoff to the acceleration, such as a linear or exponential falloff.

ACKNOWLEDGEMENTS

This work was supported by the DOD grant W911NF-16- 2-0016. We thank anonymous reviewers for their comments. We thank Lizhou Cao for his help in game and 3D user interface development. We thank Sarah Meacham and Victoria Blakley for their help in gestural data collection.

REFERENCES

[1] Shamir Alavi, Dennis Arsenault, and Anthony Whitehead. 2016. Quaternion-Based Gesture Recognition Using Wireless Wearable Motion Capture Sensors. *Sensors* 16, 5 (2016), 605. https://doi.org/10.3390/s16050605

[2] Allan Pease Barbara Pease. 2006. *The Definitive Book of Body Language: The Hidden Meaning Behind People's Gestures and Expressions*. BANTAM DELL. https://www.ebook.de/de/product/5169074/barbara_pease_allan_pease_the_definitive_book_of_body_language_the_hidden_meaning_behind_people_s_gestures_and_expressions.html

[3] John S. Bridle. 1990. Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition. In *Neurocomputing*. Vol. 68. Springer Berlin Heidelberg, 227–236. https://doi.org/10.1007/978-3-642-76153-9_28

[4] J. C. Calella, F. R. Ortega, N. Rishe, J. F. Bernal, and A. Barreto. 2016. HandMagic: Towards user interaction with inertial measuring units. In *2016 IEEE SENSORS*. 1–3. https://doi.org/10.1109/ICSENS.2016.7808524

[5] Justine Cassell. 1998. A framework for gesture generation and interpretation. *Computer vision in human-machine interaction* (1998), 191–215. https://doi.org/10.1017/CBO9780511569937.013

[6] Xiaodong Cui, Vaibhava Goel, and Brian Kingsbury. 2015. Data Augmentation for Deep Neural Network Acoustic Modeling. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.* 23, 9 (Sept. 2015), 1469–1477. https://doi.org/10.1109/TASLP.2015.2438544

[7] Youchen Du, Shenglan Liu, Lin Feng, Menghui Chen, and Jie Wu. 2017. Hand Gesture Recognition with Leap Motion. *CoRR* abs/1711.04293 (2017). arXiv:1711.04293 http://arxiv.org/abs/1711.04293

[8] L. E. Dunne, B. Smyth, and B. Caulfield. 2007. A Comparative Evaluation of Bend Sensors for Wearable Applications. In *2007 11th IEEE International Symposium on Wearable Computers*. 121–122. https://doi.org/10.1109/ISWC.2007.4373797

[9] S. Finkbeiner. 2013. MEMS for automotive and consumer electronics. In *2013 Proceedings of the ESSCIRC (ESSCIRC)*. 9–14. https://doi.org/10.1109/ESSCIRC.2013.6649059

[10] Marcus Georgi, Christoph Amma, and Tanja Schultz. 2015. Recognizing Hand and Finger Gestures with IMU Based Motion and EMG Based Muscle Activity Sensing. In *Proceedings of the International Joint Conference on Biomedical Engineering Systems and Technologies - Volume 4 (BIOSTEC 2015)*, Vol. 4. SCITEPRESS - Science and Technology Publications, Lda, Portugal, 99–108. https://doi.org/10.5220/0005276900990108

[11] P. D. Groves. 2015. Navigation using inertial sensors [Tutorial]. *IEEE Aerospace and Electronic Systems Magazine* 30, 2 (Feb 2015), 42–69. https://doi.org/10.1109/MAES.2014.130191

[12] Jeffrey Hansberger, Chao Peng, Victoria Blakely, Sarah Meacham, Lizhou Cao, and Nicholas Diliberti. 2019. *A Multimodal Interface for Virtual Information Environments*. Vol. 11574. 59–70. https://doi.org/10.1007/978-3-030-21607-8_5

[13] Jeffrey Hansberger, Chao Peng, Shannon L. Mathis, Vaidyanath Areyur Shanthakumar, Sarah C. Meacham, Lizhou Cao, and Victoria R. Blakely. 2017. Dispelling the Gorilla Arm Syndrome: The Viability of Prolonged Gesture Interactions. In *Lecture Notes in Computer Science*. Springer International Publishing, 505–520. https://doi.org/10.1007/978-3-319-57987-0_41

[14] T.C. Horton, S. Sauerland, and T.R.C. Davis. 2007. The effect of flexor digitorum profundus quadriga on grip strength. *The Journal of Hand Surgery: British & European Volume* 32, 2 (2007), 130 – 134. https://doi.org/10.1016/j.jhsb.2006.11.005

[15] E. Jeong, J. Lee, and D. Kim. 2011. Finger-gesture Recognition Glove using Velostat (ICCAS 2011). In *2011 11th International Conference on Control, Automation and Systems*. 206–210. https://ieeexplore.ieee.org/document/6106275

[16] Maria Karam and m. c. schraefel. 2005. A Taxonomy of Gestures in Human Computer Interactions. (2005). https://eprints.soton.ac.uk/261149/

[17] P. Keir, J. Payne, J. Elgoyhen, M. Horner, M. Naef, and P. Anderson. 2006. Gesture-recognition with Non-referenced Tracking. In *3D User Interfaces (3DUI'06)*. 151–158. https://doi.org/10.1109/VR.2006.64

[18] G. Drew Kessler, Larry F. Hodges, and Neff Walker. 1995. Evaluation of the CyberGlove As a Whole-hand Input Device. *ACM Trans. Comput.-Hum. Interact.* 2, 4 (Dec. 1995), 263–283. https://doi.org/10.1145/212430.212431

[19] Diederik P. Kingma and Jimmy Ba. [n. d.]. Adam: A Method for Stochastic Optimization. *3rd International Conference for Learning Representations* ([n. d.]). arXiv:cs.LG/http://arxiv.org/abs/1412.6980v9 http://arxiv.org/abs/1412.6980v9

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 1097–1105. https://dl.acm.org/citation.cfm?id=3065386

[21] Catherine E. Lang and Marc H. Schieber. 2004. Human Finger Independence: Limitations due to Passive Mechanical Coupling Versus Active Neuromuscular Control. *Journal of Neurophysiology* 92, 5 (2004), 2802–2810. https://doi.org/10.1152/jn.00480.2004 arXiv:https://doi.org/10.1152/jn.00480.2004 PMID: 15212429.

[22] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324. https://doi.org/10.1109/5.726791

[23] Scott K Liddell. 2003. *Grammar, Gesture, and Meaning in American Sign Language*. Cambridge University Press. https://doi.org/10.1017/cbo9780511615054

[24] Wei Lu, Zheng Tong, and Jinghui Chu. 2016. Dynamic Hand Gesture Recognition With Leap Motion Controller. *IEEE Signal Processing Letters* 23, 9 (Sept 2016), 1188–1192. https://doi.org/10.1109/LSP.2016.2590470

[25] Granit Luzhnica, Jorg Simon, Elisabeth Lex, and Viktoria Pammer. 2016. A sliding window approach to natural hand gesture recognition using a custom data glove. In *2016 IEEE Symposium on 3D User Interfaces (3DUI)*. IEEE, 81–90. https://doi.org/10.1109/3DUI.2016.7460035

[26] Yuntao Ma, Yuxuan Liu, Ruiyang Jin, Xingyang Yuan, Raza Sekha, Samuel Wilson, and Ravi Vaidyanathan. 2017. Hand gesture recognition with convolutional neural networks for the multimodal UAV control. In *2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*. IEEE, 198–203. https://doi.org/10.1109/RED-UAS.2017.8101666

[27] Giulio Marin, Fabio Dominio, and Pietro Zanuttigh. 2014. Hand gesture recognition with leap motion and kinect devices. In *2014 IEEE International Conference on Image Processing (ICIP)*. IEEE, 1565–1569. https://doi.org/10.1109/ICIP.2014.7025313

[28] Giulio Marin, Fabio Dominio, and Pietro Zanuttigh. 2015. Hand gesture recognition with jointly calibrated Leap Motion and depth sensor. *Multimedia Tools and Applications* 75, 22 (feb 2015), 14991–15015. https://doi.org/10.1007/s11042-015-2451-6

[29] Vinod Nair and Geoffrey E. Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning (ICML'10)*. Omnipress, USA, 807–814. http://dl.acm.org/citation.cfm?id=3104322.3104425

[30] Pedro Neto, Dario Pereira, J. Norberto Pires, and A. Paulo Moreira. 2013. Real-time and continuous hand gesture spotting: An approach based on artificial neural networks. In *2013 IEEE International Conference on Robotics and Automation*. IEEE, 178–183. https://doi.org/10.1109/icra.2013.6630573

[31] Luis Perez and Jason Wang. 2017. The Effectiveness of Data Augmentation in Image Classification using Deep Learning. *CoRR* abs/1712.04621 (2017). arXiv:1712.04621 http://arxiv.org/abs/1712.04621

[32] Rosalina, Lita Yusnita, Nur Hadisukmana, R. B. Wahyu, Rusdianto Roestam, and Yuyu Wahyu. 2017. Implementation of real-time static hand gesture recognition using artificial neural network. In *2017 4th International Conference on Computer Applications and Information Processing Technology (CAIPT)*. IEEE, 1–6. https://doi.org/10.1109/CAIPT.2017.8320692

[33] Giovanni Saggio. 2011. Electrical Resistance Profiling of Bend Sensors Adopted to Measure Spatial Arrangement of the Human Body. In *Proceedings of the 4th International Symposium on Applied Sciences in Biomedical and Communication Technologies (ISABEL '11)*. ACM, New York, NY, USA, Article 5, 5 pages. https://doi.org/10.1145/2093698.2093703

[34] Giovanni Saggio. 2012. Mechanical model of flex sensors used to sense finger movements. *Sensors and Actuators A: Physical* 185 (2012), 53 – 58. https://doi.org/10.1016/j.sna.2012.07.023

[35] Thomas Schlömer, Benjamin Poppinga, Niels Henze, and Susanne Boll. 2008. Gesture Recognition with a Wii Controller. In *Proceedings of the 2Nd International Conference on Tangible and Embedded Interaction (TEI '08)*. ACM, New York, NY, USA, 11–14. https://doi.org/10.1145/1347390.1347395

[36] Zhong Shen, Juan Yi, Xiaodong Li, Mark Hin Pei Lo, Michael Z Q Chen, Yong Hu, and Zheng Wang. 2016. A soft stretchable bending sensor and data glove applications. *Robotics and Biomimetics* 3, 1 (2016), 22. https://doi.org/10.1186/s40638-016-0051-1

[37] Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *ICLR 2015* (2014). arXiv:cs.CV/http://arxiv.org/abs/1409.1556v6 https://arxiv.org/abs/1409.1556

[38] Yale Song, David Demirdjian, and Randall Davis. 2012. Continuous Body and Hand Gesture Recognition for Natural Human-computer Interaction. *ACM Trans. Interact. Intell. Syst.* 2, 1, Article 5 (March 2012), 28 pages. https://doi.org/10.1145/2133366.2133371

[39] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going Deeper with Convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2014-09-17). IEEE. https://doi.org/10.1109/CVPR.2015.7298594 arXiv:cs.CV/http://arxiv.org/abs/1409.4842v1

[40] Chong Wang, Zhong Liu, and Shing-Chow Chan. 2015. Superpixel-Based Hand Gesture Recognition With Kinect Depth Camera. *IEEE Transactions on Multimedia* 17, 1 (Jan 2015), 29–39. https://doi.org/10.1109/TMM.2014.2374357

[41] Yingying Wang and Michael Neff. 2013. Data-driven Glove Calibration for Hand Motion Capture. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '13)*. ACM, New York, NY, USA, 15–24. https://doi.org/10.1145/2485895.2485901

[42] Deyou Xu. 2006. A Neural Network Approach for Hand Gesture Recognition in Virtual Reality Driving Training System of SPG. In *18th International Conference on Pattern Recognition (ICPR'06)*, Vol. 3. IEEE, 519–522. https://doi.org/10.1109/ICPR.2006.109

[43] F Yazadi. 2009. Cyberglove systems cyberglove ii wireless data glove user guide. *CyberGlove Systems LLC* (2009). https://static1.squarespace.com/static/559c381ee4b0ff7423b6b6a4/t/574f4c35b654f98f724d1927/1464814655198/CyberGloveII_UserGuide_2009_0.pdf