applied sciences

MDPI

Article

Real-Time Hand Gesture Spotting and Recognition Using RGB-D Camera and 3D Convolutional Neural Network

Dinh-Son Tran, Ngoc-Huynh Ho, Hyung-Jeong Yang *, Eu-Tteum Baek, Soo-Hyung Kim and Gueesang Lee

School of Electronics and Computer Engineering, Chonnam National University, 77 Yongbong-ro, Gwangju 500-757, Korea; trandinhson3086@gmail.com (D.-S.T.); 176276@jnu.ac.kr (N.-H.H.); geodo100@gmail.com (E.-T.B.); shkim@jnu.ac.kr (S.-H.K.); gslee@jnu.ac.kr (G.L.)

* Correspondence: hjyang@jnu.ac.kr

Received: 17 December 2019; Accepted: 16 January 2020; Published: 20 January 2020

Abstract: Using hand gestures is a natural method of interaction between humans and computers. We use gestures to express meaning and thoughts in our everyday conversations. Gesture-based interfaces are used in many applications in a variety of fields, such as smartphones, televisions (TVs), video gaming, and so on. With advancements in technology, hand gesture recognition is becoming an increasingly promising and attractive technique in human–computer interaction. In this paper, we propose a novel method for fingertip detection and hand gesture recognition in real-time using an RGB-D camera and a 3D convolution neural network (3DCNN). This system can accurately and robustly extract fingertip locations and recognize gestures in real-time. We demonstrate the accurateness and robustness of the interface by evaluating hand gesture recognition across a variety of gestures. In addition, we develop a tool to manipulate computer programs to show the possibility of using hand gesture recognition. The experimental results showed that our system has a high level of accuracy of hand gesture recognition. This is thus considered to be a good approach to a gesture-based interface for human–computer interaction by hand in the future.

Keywords: hand gesture spotting and recognition; 3DCNN; human–computer interaction

1. Introduction

At present, with constant developments in information technology, human beings are attempting to communicate with computers in more natural ways. The traditional input devices for human–computer interaction (HCI), such as mice, keyboards, remotes, and so on, no longer serve as natural means of interaction owing to their lack of flexibility. In general, the natural ways for humans to interact with computers include voice commands and body language, and these are available in many commercial electronic products. Although speech recognition is one of the natural methods for human–computer interaction, it is limited by noisy environments and the different ways that people pronounce the same words. Another human–computer communication method involves body language-based interaction. In many cases, interacting with a computer through body language is more reliable than other methods. Body language can involve different types of expression, such as hand gestures, body poses, and facial emotions. Compared with gestures with other body parts, hand gestures are the most efficient means of meaningful expression and serve as a common language for all humans. For example, waving one's hand to express 'hello' is pretty flexible.

Hand gesture recognition is one method of interaction for human beings. Hand gestures are defined as movements of the human hand that are used to express meaningful information and

thoughts. We use our hands every day while talking to present ideas, points at people or things, and so on. Everyone is familiar with hand gesture interaction. It has even been suggested that hand gestures will soon replace the touchscreen technology of mobile phones and other devices. In HCI systems, hand gesture-based interfaces are broadly applied in many practical applications, such as sign language recognition [1], robot control [2], virtual mouse control [3–8], exploration of medical image data [9], human–vehicle interaction [10], and immersive gaming technology [11]. Therefore, the recognition of human hand gestures from videos has been one of the main goals of computer vision for decades, particularly with traditional cameras (RGB cameras).

However, even when using RGB cameras, most existing algorithms tend to fail when faced with changing light levels, complex backgrounds, different user–camera distances, multiple people, or background or foreground movements during the hand tracking. Microsoft's Kinect RGB with depth (RGB-D) camera has extended depth-sensing technology and interfaces for human-motion analysis applications.

More recently, systems using convolutional neural networks (CNNs) have shown outstanding performance in HCI [12]. Owing to its use of multiple layers, CNNs' architecture is built for automating feature learning. Therefore, through the use of a deep learning framework, data are represented by a hierarchy of features arranged from low-level to high-level.

In this paper, the main goal of our system is to discover a hand gesture-based interface from depth videos. We present a combination of fingertip detection and a 3D convolutional neural network for hand gesture recognition from a depth camera. The proposed network architecture effectively exploits multi-level features to generate high-performance predictions.

The remainder of the paper is organized as follows. In the second section, we present our proposed approach and the related materials that we used in this paper. The architecture of the proposed method is then discussed in Section 3. Comprehensive experiments on a challenging dataset are used to validate the effectiveness of our system in Section 4. Finally, in Section 5, the conclusion and future works are presented.

2. Related Works

This investigation is organized as follows. Section 2.1 reviews the sensors used for the gestural interface and Section 2.2 surveys the hand gesture recognition systems. In Section 2.3, a review of convolutional neural networks can be found.

2.1. Sensors Used for Hand Gesture Recognition Interface

Typically, there are two main types of sensors used for hand gesture recognition: gloves-based sensors and vision-based sensors. The gloves-based techniques require users to wear a hand glove fitted with sensors for recognizing the hand posture. By contrast, the vision-based approaches require only a camera, consequently achieving natural communication in human–computer interaction without the need for any extra devices.

Many previous studies on hand gesture recognition have been conducted using gloves fitted with sensors, such as the works of [13–19]. However, despite some remarkable successes, this method remains challenging owing to its complexity as well as variable glove sizes between users. Consequently, many recent efforts have been focused on vision-based systems. These methods require no gloves or hardware except for a sensor, so these systems are substantially cheaper and can be easily accepted by any field.

Vision-based sensors can be categorized into two types, namely the traditional camera (RGB)-based systems and depth sensor (RGB-D)-based systems, respectively. Using traditional cameras, the model-based approaches [7,20–28] generate hand model hypotheses and evaluate them with the available visual observations. These approaches exhibited clear detection difficulty when the light levels were changed or a complex background was used, and also required a fixed distance from the camera to the users. In order to overcome these limitations, some studies have used depth cameras, for

example, PrimeSense, AsusensXtion Pro, and Microsoft's Kinect [29]. These cameras have advanced significantly over the past few years, and their performances have increased, while their prices have lowered. Compared with traditional RGB cameras, depth cameras offer many advantages: 30 frames per second with depth resolution, working in low light levels, and tracking at a longer distance. Some systems use depth images from Kinect and achieve high speeds, while avoiding the disadvantages of traditional RGB cameras by tracking depth maps from frame to frame [30–32]. These methods use a complex mesh model and achieve real-time performance. However, such systems only work for hand tracking, not hand gesture recognition such as swiping left, right, up, down, or zoom in and out, and so on.

2.2. Hand Gesture Recognition Using RGB-D Sensor

Fingertip-based hand gesture recognition is currently one of the interesting challenges in computer vision, owing to its applications in many different areas such as virtual mice, remote controls, sign-language recognition, and immersive gaming technology. The K-curvature is a famous algorithm used to detect the certain shape of an object [33] as well as for fingertip detection. One fingertip-based hand gesture application was developed using the K-curvature method and Microsoft's Kinect sensor [34]. However, this method showed some limitations regarding hand gestures, specifically with the accuracy of the gesture detection and the fact that the user only uses one hand to perform in front of the camera without using their body. Duong et al. [35] implemented a method for fingertip detection using RGB-D images and convolution neural networks. This method showed promising hand detection results in terms of accuracy and a new direction for fingertip detection using an RGB-D camera. In this way, depth-based systems avoid all of the problems mentioned above associated with RGB-cameras. However, it did not perform hand gesture recognition or precise recognition.

Hand gesture recognition is commonly used to control a computer interface that is in the near vicinity of the user. The hand is first found in the input image, and features are extracted, most often based on shape. In the field of HCI, many systems using depth cameras have been developed for computer control using hand gestures [5,21,29,32,36,37]. In summary, the main challenge facing hand gesture recognition from depth cameras is the lack of a dataset that contains both fingertip information and hand gesture recognition. Therefore, it is difficult to achieve high accuracy and flexibility in real-time using the current systems.

One system with a goal similar to ours can be found in the work of [38]. With the same seven gestures, using Kinect V2 and a convolutional neural network, this system achieved real-time performance with high accuracy on the 20BN-jester dataset V1 [39]. However, the original videos used were only 12 frames per second, which seems slow compared with real-time systems. Further, the system was not detecting the fingertips.

2.3. 3D Convolutional Neural Networks

In the past few years, convolutional neural networks (CNNs) have achieved state-of-the-art performance in image content analysis and classification tasks. From AlexNet [40], VGG [41], GoogleNet [42], to ResNet [43], all of these deep CNN approaches have achieved outstanding success in image understanding. However, for hand gesture recognition in videos, these approaches showed drawbacks in that they have not only huge computational cost, but they also fail to capture long context with multi-frames and high-level information. Three-dimensional convolutional neural networks (3DCNNs) [44] have been proposed and shown to achieve the best performance for human action recognition from video on the recognition of human actions dataset [45]. More recently, 3DCNNs have shown strong performance on various tasks when trained on large-scale datasets [46], including action detection [47], hand gesture detection [48], and video captioning [49].

Ensemble models often provide higher accuracy of prediction than single models [50]. Inspired by ensemble models, in this paper, we present the use of multiple three-dimensional convolutional neural networks on hand gesture recognition problems.

3. Proposed Method

An overview of the proposed system is presented in Figure 1. The hand region of interest is first extracted using in-depth skeleton-joint information images from a Microsoft Kinect Sensor version 2, and the contours of the hands are extracted and described using a border-tracing algorithm. The K-cosine algorithm is used to detect the fingertip location based on the hand-contour coordinates model, and the result of fingertip detection is transformed into the gesture initialization in order to spot hand gestures. Finally, a gesture is recognized based on the 3D convolutional neural network.

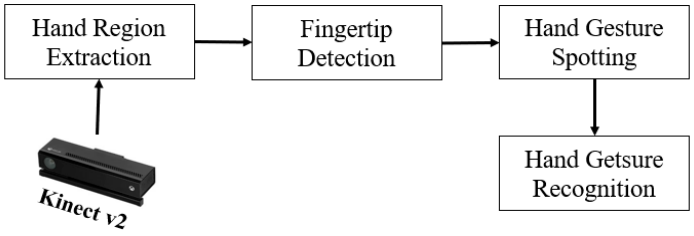


Figure 1. Algorithm pipeline.

3.1. Hand Region Extraction

In this section, we describe the hand region extraction step, as shown in Figure 2. The hand region of interest and the center of the palm are first extracted from depth images provided by the Kinect V2 skeletal tracker, and then converted to binary images. The hand contours are extracted and described using a border-tracing algorithm.

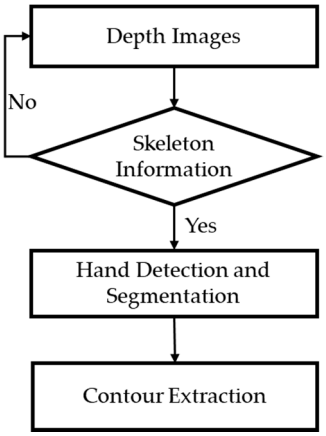


Figure 2. Flowchart of hand region extraction.

The depth images used to detect the hand based on the Kinect V2 skeletal tracker are shown in Figure 5a. These images are captured using a Microsoft Kinect V2 sensor, which estimates each of the user's body parts through input depth images and maps the learned body parts to the depth images through various user actions. In this manner, the camera can obtain skeleton-joint information for 25 joints, as shown in Figure 3, for example, hip, spine, head, shoulder, hand, foot, and thumb. Using the depth image obtained by the Kinect skeletal tracker, the hand region of interest (HRI) and the center of the palm are easily and effectively extracted.

In order to remove noise from the hand region, the median filter and morphological processing [51] are applied. The results are then exported to the binary image, based on Kinect's depth signals with fixed thresholds.

Once the binary images of the hand regions are detected, the hand contours are computed using the Moore–Neighbor algorithm [52]. This method is one of the most common algorithms used to extract the contours of objects (regions) from an image. At the end of this process, we obtain the

contour pixels of the hand as a boundary point set, $P_0, \ldots, P_{i-1}, P_i, \ldots, P_n$. These values are used to measure the curvature.

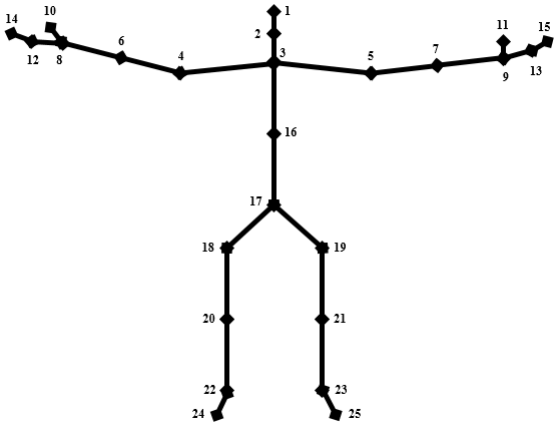


Figure 3. Skeletal joint information based on the Kinect skeletal tracker.

3.2. Fingertip Detection

After extracting the hand contour, the K-cosine corner detection [33] algorithm computes the fingertip points based on the coordinates of the detected hand contours. This is a well-known algorithm used to detect the shapes of certain objects, and it is also used in fingertip detection. It attempts to determine the angle between the vectors of a finger, as shown in Figure 4.

$$\cos \alpha_i = \cos(\vec{a_{ik}}, \vec{b_{ik}}) = \frac{\vec{a_{ik}} \cdot \vec{b_{ik}}}{\left|\vec{a_{ik}}\right|\left|\vec{b_{ik}}\right|} \tag{1}$$

Equation (1) is used to determine the fingertip locations, where i is a boundary point and k is a constant value that obtained from the experiment. In this calculation, 10 is the best choice for value k and is suitable for almost all situations. The k-vectors at the point $P_i(x_i, y_i)$ are $\vec{a_{ik}} = (x_i - x_{i+k}, y_i - y_{i+k})$ and $\vec{b_{ik}} = (x_i - x_{i-k}, y_i - y_{i-k})$. $\cos \alpha_i$ denotes the cosine of angle between $\vec{a_{ik}}$ and $\vec{b_{ik}}$ for a given pixel P_i.

The angle α_i obtained by the K-cosine algorithm, is used to discriminate between the fingertips and the valleys as a threshold. The fingertips are considered as a positive curvature, while the valleys are a negative curvature. If the point value of $\cos \alpha_i$ is the greater or equal to threshold, it is considered to be the fingertip candidate. This process results in sets of pixels belonging to the hands and fingertip locations, as shown in Figure 5b.

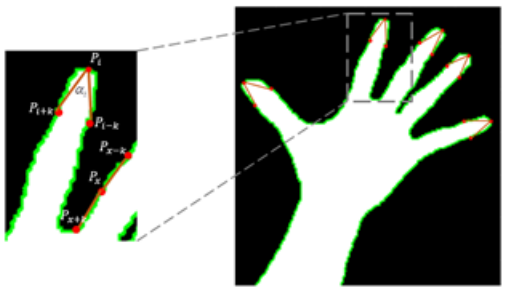


Figure 4. Fingertip detection using the K-cosine algorithm.

(**a**) (**b**)

Figure 5. Results of fingertip detection: (**a**) depth images, (**b**) fingertip detection.

3.3. Hand Gesture Spotting

3.3.1. Target Person Locking

When multiple people are present, the targeted person is the one chosen to perform hand gesture during tracking. In this work, the Kinect V2 sensor can extract skeleton information for up to six people at once. Therefore, using the algorithm described above, the system can locate the fingertips of up to six people. However, in order to recognize the hand gesture, we need to identify the target person so as to eliminate the influence of the others. To accomplish this, we used a user-locking algorithm to solve the problem during hand tracking. The implementation is presented in Algorithm 1.

In this algorithm, the given detected head joint coordinates and right-hand joint coordinates of multi-people in-depth image using the Kinect skeletal tracker, as shown in Figure 3, we define the target person based on the distance head-hand. The user who raises his hand over his head in 10 frames is the target user.

Algorithm 1 Target Person Locking

begin
Require User ID, head joint coordinates (x1, y1), right hand joint coordinates (x13, y13).
1: **Calculation** the distance between y1 and y13.
2: **If** y13 > y1 during 10 frames
then process target user
else redo 1
end begin

3.3.2. Hand Gesture Spotting

In real-time hand gesture recognition, it is still challenging to determine when a gesture begins and when it ends from a hand trajectory. This session presents a new solution for hand gesture spotting to mark the start and end points of a gesture, as shown in Figure 6.

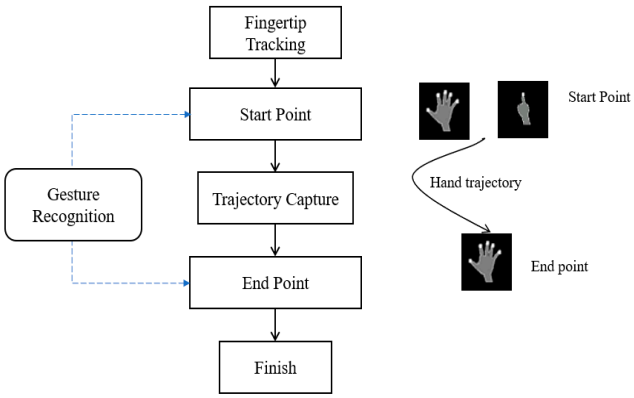


Figure 6. Proposed scheme for gesture spotting.

In this step, the depth resolution is obtained from the Kinect V2 sensor as 512 × 424 pixels, and we define six hand gestures of swiping left (SL), swiping right (SR), swiping down (SD), swiping up (SU), zoom in (ZI), and zoom out (ZO) as six different gesture trajectories. A remaining gesture is meaningless (M), which is defined as a free trajectory.

During tracking, the hand trajectory is recognized in every frame using the methods mentioned in Section 3.4. Figure 6 shows the proposed hand gesture spotting scheme of our system.

3.4. Hand Gesture Recognition

We exploit 3DCNN to recognize hand gestures by using the better capturing of the spatio-temporal information in videos. Compared with the two-dimensional convolutional neural networks (2DCNNs), the 3DCNN model has one advantage, that it captures motion information by applying the convolution operation in both time and space. The inputs are full frames of videos and do not rely on any pre-processing, so the model can easily be extended to large datasets. The details of the 3DCNN architecture are shown in Figure 7.

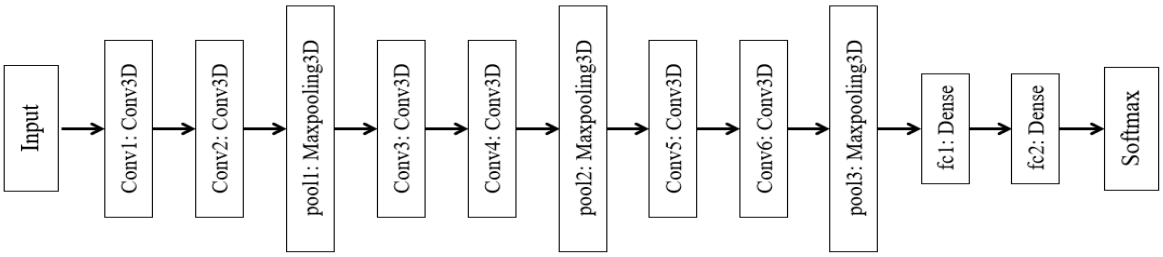


Figure 7. Block diagram of the three-dimensional convolutional neural network (3DCNN).

The network has six convolutional layers (Conv1, Conv2, Conv3, Conv4, Conv5, and Conv6), three max-pooling layers (pool1, pool2, and pool3), and two fully connected layers (fc1, fc2) before softmax output. The fully connected layer has 512 nodes with 50% dropout. Each input is a human action video. The input dimensions are $d \times h \times w$, where d is the length of the normalized video inputs, and h and w are the height and width of the frame, respectively. Because the lengths of the frame numbers in the videos of the dataset are different, so as to normalize the temporal lengths of the videos, we down-sampled to d frames per video. In our architecture, $d = 20$ is the optimal number, and matches the capabilities of the computer. The 20 frames are chosen to spread from the beginning to end of videos using the following formula.

$$d_i = \left\lfloor \frac{i * N}{20} \right\rfloor, \quad i = 1, 2, \ldots, 20, \tag{2}$$

where $d_i \in d$ is the ith frame chosen from the N frames of original video, for real z; $\lfloor z \rfloor$ is a floor function [53], which returns the greatest integer less than or equal to z; and N is the number of original input frames. We also re-sampled d frames to 32 × 32 ($w \times h$) per video.

As shown in Figure 7, the proposed network has six layers of 3D convolutions, with each pair of convolutions followed by a max-pooling, and two fully connected layers. The filter numbers of Conv1, Conv2, Conv3, Conv4, Conv5, and Conv6 are 32, 32, 64, 64, 128, and 128, respectively. In each convolution layers, the size of the kernels is 3 × 3 × 3, where the first number 3 is in the temporal dimension and the remaining 3 × 3 is in the spatial dimension. The pool1, pool2, and pool3 are max-pooling following each pair of convolutions, and the size of fc1 and fc2 is 512 nodes. In particular, we add a dropout layer between the convolution layer and the max-pooling layer so as to eliminate overfitting. The output layer activation is implemented with softmax.

In order to improve the accuracy of the model, we use an ensemble model of 3DCNN that predicts the class labels based on the predicted probabilities for classifiers, as shown in Figure 8. 3DCNN_1, 3DCNN_2, and 3DCNN_n are models 1, 2, and n in the n-model ensemble, respectively. Each model

has a different weight volume. In order to aggregate the results of each individual 3DCNN, we use the ensemble output to obtain a final probability. The optimal weighting of the final softmax, the maximum of averaging probability, is the output of the final prediction. The formula of the 3DCNN Ensemble procedure is as follows:

$$\hat{y} = \underset{j}{\operatorname{argmax}} \left(\frac{1}{n} \sum_{i=1}^{n} p_{ij} \right). \tag{3}$$

In Equation (3), p_{ij} is the predicted probabilities p of label jth of the ith classifier, n is the number of models used for the learning ensemble, and $\hat{y}$ is the final prediction. The outputs of all single models in Figure 8 are combined to form the final prediction by optimal.

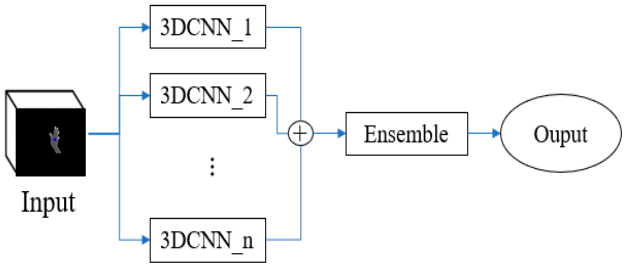


Figure 8. Ensemble learning of 3DCNN.

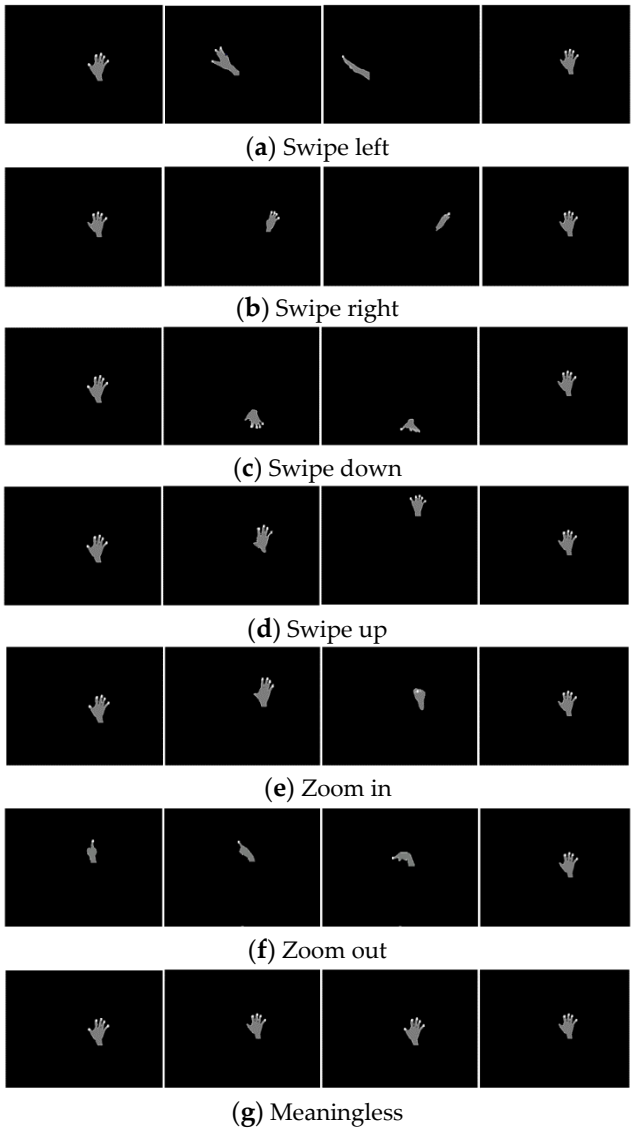
4. Experiments

4.1. Dataset

In this work, a hand gesture dataset was prepared for conducting experiments to validate the proposed methodology with manually labeled ground truth. The dataset contains videos of hand gestures using fingertip detection from depth videos. It contains videos of seven hand gestures: SL, SR, SD, SU, ZI, ZO, and M. Each gesture is performed 15 times by 50 participants, resulting in 5250 videos in total, as shown. All of the participants were right-handed, as we focused on right-hand movement for simplicity and accurate detection. These videos were collected in three places: the laboratory, the corridor, and the classroom. The experimental setup is shown in Figure 9.

Figure 9. Recording dataset.

The dataset also included many different cases, for example, normal lighting and faint lighting, different backgrounds, and changing the user–camera distance from 0.5 m to 4 m. The speed of the tracking process was 30 frames per second, with the lengths of sample gestures ranging from 20 frames to 45 frames. Figure 10 shows examples of the hand gesture dataset for the proposed system.



(**a**) Swipe left

(**b**) Swipe right

(**c**) Swipe down

(**d**) Swipe up

(**e**) Zoom in

(**f**) Zoom out

(**g**) Meaningless

Figure 10. Examples from dataset.

4.2. Implementation Details

We developed the proposed system on a desktop PC with an AMD Ryzen 5 1600 Six-Core Processor 3.20 GHz CPU with 16 GB of RAM and a GeForce GTX 1080 Ti GPU. The system was implemented within the C# and Python programing languages. The speed of the tracking process was 30 frames per second. The Keras, Scikit-learn 0.20 frameworks of deep learning were used to implement the 3DCNN model, as described in Section 3.4.

4.3. Evaluation

In the experiment for hand gesture recognition, we randomly select 70% original data as training data, 10% as validation data, and the remaining 20% as testing data. In order to evaluate our proposed method for hand gesture recognition in videos, we compare the proposed model with the baseline method: a traditional machine learning model—support vector machine (SVM) [45]—and the two-dimensional convolution neural network (2DCNN). The 2DCNN follows the same network as the 3DCNN version, but only with 2D convolutions instead of 3D convolutions. The input of the traditional ML model (SVM) was the same as the input of a 2DCNN. SVM nonlinear multiclass algorithm with histogram intersection kernel was implemented using library Scikit-learn 0.20.

Three metrics were used to compare the performance of the algorithms. First, the accuracy was introduced to measure when there is an equal number of samples belonging to each class. It is the ratio of number of correct predictions to the total number of input samples. The second and third metrics were the training time and testing time of the deep learning model. Training time is considered a scenario of online training while collecting data. Each training session was ended if the test classification accuracy of the validation datasets did not change over 100 training epochs. To calculate the testing time, the paper was conducted five times, and the average of the results was used.

4.4. Results

Figure 11 illustrates the accuracy of the classification of 3DCNN model on the train and validation sets. The plots suggest that the model has a good fit on the problem.

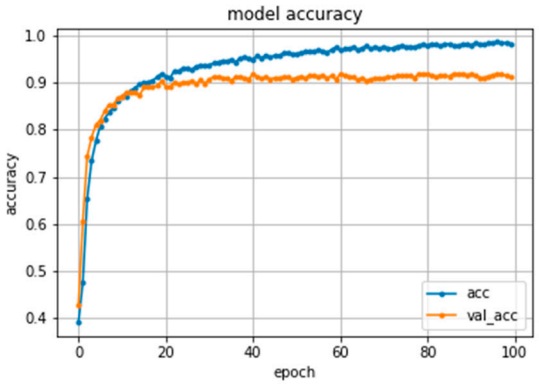


Figure 11. The accuracy of 3DCNN model during training.

We compare our results on the hand gesture dataset against another state-of-the-art method in Table 1. Overall, DL algorithms showed higher accuracy than the traditional ML model. The 3DCNN model achieves 92.6% accuracy, while SVM and CNN are 60.50% and 64.28%, respectively, which are 32.1% and 28.3% lower than the 3DCNN result. The results showed that the 3DCNN network outperforms SVM and 2DCNN in video classification on our hand gesture dataset. One reason for this result is that learning multiple frames in 3DCNN provide the flexibility of a decision boundary.

Table 1. Comparison results with support vector machine (SVM) and two-dimensional convolution neural network (2DCNN).

Models	Training Time	Testing Time	Accuracy (%)
SVM	21.94 m	2.57 s	60.50
2DCNN	50.00 m	3.46 s	64.28
3DCNN	1h35	5.29 s	92.60

In terms of the training time and testing time of convolutional neural networks, the training time of the 3DCNN showed higher run-time than SVM and 2DCNN. However, all architectures could recognize the hand gesture in less than 5.29 s, which shows the feasibility of implementing neural networks to real-time systems.

Table 2 shows the results of hand gesture recognition by our proposed method when using ensemble learning. It can easily be seen that the ensemble learning method outperforms the single 3DCNN in terms of video classification with an ensemble of 15 3DCNN models, which achieves 97.12% accuracy. The experimental results indicate that the 3DCNN ensemble with five models is substantially more accurate than the individual 3DCNN on the dataset. However, when the number of models increases to 10 and 15, there is no significant change. On the other hand, ensemble learning with many

models also required a long training time, which is related to the expensive computational cost of a computer.

Table 2. Experimental result of the ensemble learning approach.

Models	Accuracy (%)
3DCNN	92.60
Ensemble model with 5 different 3DCNN networks	96.42
Ensemble model with 10 different 3DCNN networks	96.82
Ensemble model with 15 different 3DCNN networks	97.12

The confusion matrix of the 3DCNN with 15 models ensemble is shown in Table 3. As expected, the highest accuracy was obtained in the easier gestures of 'swipe left (SL)', 'swipe right (SR)', and 'meaningless (M)', while the lowest accuracy was obtained in the harder gestures 'zoom in (ZI)' and 'zoom out (ZO)'. The accuracy was reduced in the 'drag' gesture because, with the starting point and ending point, the gesture was sometimes confused with others owing to fast hand movement.

Table 3. Confusion matrix of the proposed method. SL, swipe left; SR, swipe right; SU, swipe up; SD, swipe down; ZI, zoom in; ZO, zoom out; M, meaningless.

Target	Selected							Acc
	SL	SR	SU	SD	ZI	ZO	M	
SL	149	0	0	0	0	0	1	0.99
SR	1	148	0	0	0	0	1	0.99
SU	1	0	145	0	2	0	2	0.97
SD	0	0	0	146	1	1	2	0.97
ZI	0	0	0	0	143	4	3	0.95
ZO	0	0	1	0	4	141	4	0.94
M	1	0	0	0	0	1	148	0.99
								0.97

4.5. Application of the Proposed System

We want to show through the application that a hand gesture-based system is available on behalf of a keyboard or mouse-based system. Thus, we develop practical application with hand gesture recognition as shown in Figure 12. When the user stands in front of the RGB-D camera, video frames are captured and the hand is detected. When the user interacts with a computer, a deep learning technique is used for hand gesture recognition internally in the system. Therefore, users can easily use the hand gestures to interact and control applications. Figure 13 shows an example of how our hand gestures-based real-time HCI system controls the computer with Kinect V2.

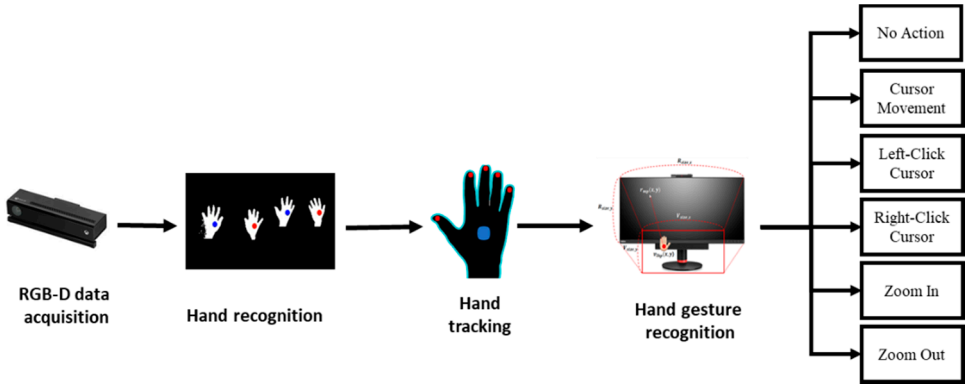
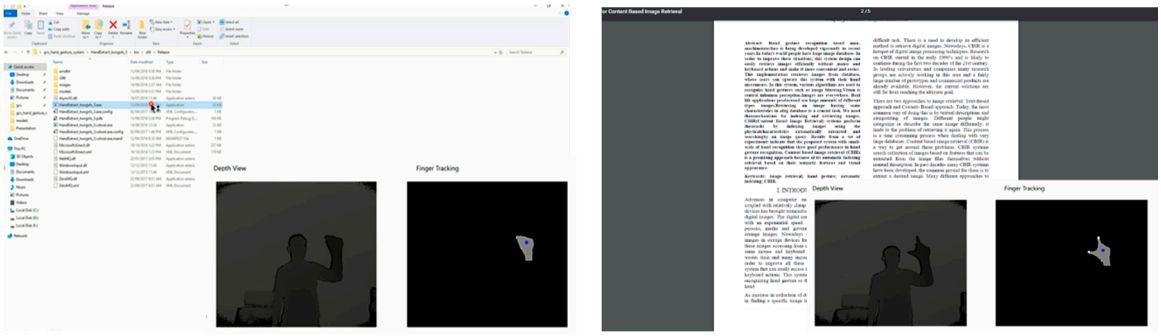


Figure 12. Flowchart of virtual mouse system.



(**a**) (**b**)

Figure 13. Examples of virtual mouse system: (**a**) left-click; (**b**) zoom-in.

5. Discussions

This paper presented a new approach to hand gesture recognition using a combination of geometry algorithm and a deep learning method to achieve fingertip detection and gesture recognition tasks. This approach exhibited not only highly accurate gesture estimates, but also suitability for practical applications. The proposed method has many advantages, for example, working well in changing light levels or with complex backgrounds, accurate detection of hand gestures at a longer distance, and recognizing the hand gestures of multiple people. The experimental results indicated that this approach is a promising technique for hand-gesture-based interfaces in real time. For future work with hand gesture recognition, we intend to expand our system to handle more hand gestures and apply our method in more other practical applications.

Author Contributions: Conceptualization, D.-S.T., N.-H.H., H.-J.Y. and E.-T.B.; Funding acquisition, S.-H.K.; Investigation, D.-S.T.; Methodology, D.-S.T.; Project administration, S.-H.K. and G.L.; Supervision, H.-J.Y.; Validation, D.-S.T.; Writing—original draft, D.-S.T.; Writing—review & editing, N.-H.H. and E.-T.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2018R1D1A3B05049058 & NRF-2017R1A4A1015559).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Starner, T.; Pentland, A. Real-time american sign language recognition from video using hidden markov models. In *Motion-Based Recognition*; Springer: Berlin/Heidelberg, Germany, 1997; pp. 227–243.
2. Malima, A.K.; Özgür, E.; Çetin, M. A Fast Algorithm for Vision-Based Hand Gesture Recognition for Robot Control. In Proceedings of the 2006 IEEE 14th Signal Processing and Communications Applications, Antalya, Turkey, 17–19 April 2006.
3. Tsai, T.-H.; Huang, C.-C.; Zhang, K.-L. Embedded Virtual Mouse System by Using Hand Gesture Recognition. In Proceedings of the 2015 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW), Taipei, Taiwan, 6–8 June 2015; pp. 352–353.
4. Kadam, S.; Sharma, N.; Shetty, T.; Divekar, R. Mouse Operations using Finger Tracking. *Int. J. Comput. Appl.* **2015**, *116*, 20. [CrossRef]
5. Jeon, C.; Kwon, O.-J.; Shin, D.; Shin, D. Hand-Mouse Interface Using Virtual Monitor Concept for Natural Interaction. *IEEE Access* **2017**, *5*, 25181–25188. [CrossRef]
6. Abhilash, S.S.; Lisho Thomas, N.W.C.C. Virtual Mouse Using Hand Gesture. *Int. Res. J. Eng. Technol.* **2018**, *5*, 4, eISSN 2395-0056, pISSN 2395-0072.
7. Le, P.D.; Nguyen, V.H. Remote mouse control using fingertip tracking technique. In *AETA 2013: Recent Advances in Electrical Engineering and Related Sciences*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 467–476.

8. Reza, M.N.; Hossain, M.S.; Ahmad, M. Real Time Mouse Cursor Control Based on Bare Finger Movement Using Webcam to Improve HCI. In Proceedings of the 2015 International Conference on Electrical Engineering and Information Communication Technology (ICEEICT), Dhaka, Bangladesh, 21–23 May 2015; pp. 1–5.
9. Gallo, L.; Placitelli, A.P.; Ciampi, M. Controller-Free Exploration of Medical Image Data: Experiencing the Kinect. In Proceedings of the 2011 24th International Symposium on Computer-Based Medical Systems (CBMS), Bristol, UK, 27–30 June 2011; pp. 1–6.
10. Dong, G.; Yan, Y.; Xie, M. Vision-Based Hand Gesture Recognition for Human-Vehicle Interaction. In Proceedings of the International Conference on Control, Automation and Computer Vision, Singapore, 18–21 November 2018.
11. Zhang, X.; Chen, X.; Wang, W.; Yang, J.; Lantz, V.; Wang, K. Hand Gesture Recognition and Virtual Game Control Based on 3D Accelerometer and EMG Sensors. In Proceedings of the 14th International Conference on Intelligent User Interfaces, Sanibel Island, FL, USA, 25 September 2009; pp. 401–406.
12. Deng, L. A tutorial survey of architectures, algorithms, and applications for deep learning. *APSIPA Trans. Signal Inf. Process.* **2014**, *3*. [CrossRef]
13. Wang, Y.; Neff, M. Data-driven Glove Calibration for Hand Motion Capture. In Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation, Anaheim, CA, USA, 26–27 July 2013; pp. 15–24.
14. Parab, P.; Kinalekar, S.; Chavan, R.; Sharan, D.; Deshpande, S. Hand Gesture Recognition using Microcontroller Flex Sensor. *Int. J. Sci. Res. Educ.* **2014**, *2*, 3.
15. Parvini, F.; McLeod, D.; Shahabi, C.; Navai, B.; Zali, B.; Ghandeharizadeh, S. An Approach to Glove-Based Gesture Recognition. In Proceedings of the International Conference on Human-Computer Interaction, San Diego, CA, USA, 19–24 July 2009; pp. 236–245.
16. Allevard, T.; Benoit, E.; Foulloy, L. Fuzzy Glove for Gesture Recognition. In Proceedings of the XVII IMEKO World Congress, Dubrovnik, Croatia, 22–28 June 2003; pp. 2026–2031.
17. Ghunawat, M.R. Multi-Point Gesture Recognition Using LED Gloves for Interactive HCI. *Int. J. Comput. Sci. Inf. Technol.* **2014**, *5*, 6768–6773.
18. Ganzeboom, M. How Hand Gestures Are Recognized Using a Dataglove. 2009. Available online: https://pdfs.semanticscholar.org/bd6b/40dca3813367272c917e6d28a45a2f053004.pdf?_ga=2.35948259.294260165.1579427347-803309327.1579427347 (accessed on 12 December 2018).
19. Vardhan, D.V.; Prasad, P.P. Hand gesture recognition application for physically disabled people. *Int. J. Sci. Res.* **2014**, *3*, 765–769.
20. Rautaray, S.S.; Agrawal, A. Real time hand gesture recognition system for dynamic applications. *Int. J. UbiComp* **2012**, *3*, 21. [CrossRef]
21. Murugeswari, M.; Veluchamy, S. Hand Gesture Recognition System for Real-Time Application. In Proceedings of the 2014 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT), Ramanathapuram, India, 8–10 May 2014; pp. 1220–1225.
22. Haria, A.; Subramanian, A.; Asokkumar, N.; Poddar, S.; Nayak, J.S. Hand gesture recognition for human computer interaction. *Procedia Comput. Sci.* **2017**, *115*, 367–374. [CrossRef]
23. Chen, Z.; Kim, J.-T.; Liang, J.; Zhang, J.; Yuan, Y.-B. Real-time hand gesture recognition using finger segmentation. *Sci. World J.* **2014**, *2014*. [CrossRef] [PubMed]
24. Xu, P. A Real-time Hand Gesture Recognition and Human-Computer Interaction System. *arXiv* **2017**, arXiv:1704.07296.
25. Neto, P.; Pereira, D.; Pires, J.N.; Moreira, A.P. Real-Time and Continuous Hand Gesture Spotting: An Approach Based on Artificial Neural Networks. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany, 6–10 May 2013; pp. 178–183.
26. Banerjee, A.; Ghosh, A.; Bharadwaj, K.; Saikia, H. Mouse control using a web camera based on colour detection. *arXiv* **2014**, arXiv:1403.4722.
27. Ge, L.; Ren, Z.; Li, Y.; Xue, Z.; Wang, Y.; Cai, J.; Yuan, J. 3D Hand Shape and Pose Estimation from a Single RGB Image. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 10833–10842.
28. Li, Y.; Xue, Z.; Wang, Y.; Ge, L.; Ren, Z.; Rodriguez, J. End-to-End 3D Hand Pose Estimation from Stereo Cameras. Available online: https://bmvc2019.org/wp-content/uploads/papers/0219-paper.pdf (accessed on 25 December 2019).

29. Fossati, A.; Gall, J.; Grabner, H.; Ren, X.; Konolige, K. *Consumer Depth Cameras for Computer Vision: Research Topics and Applications*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2012.
30. Sharp, T.; Keskin, C.; Robertson, D.; Taylor, J.; Shotton, J.; Kim, D.; Rhemann, C.; Leichter, I.; Vinnikov, A.; Wei, Y.; et al. Accurate, Robust, and Flexible Real-Time Hand Tracking. In Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, Seoul, Korea, 18–23 April 2015; pp. 3633–3642.
31. Khamis, S.; Taylor, J.; Shotton, J.; Keskin, C.; Izadi, S.; Fitzgibbon, A. Learning an Efficient Model of Hand Shape Variation from Depth Images. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 2540–2548.
32. Oikonomidis, I.; Kyriazis, N.; Argyros, A.A. Efficient Model-Based 3D Tracking of Hand Articulations Using Kinect. In Proceedings of the BmVC, Dundee, UK, 29 August–2 September 2011; Volume 1, p. 3.
33. Sun, T.-H. K-Cosine Corner Detection. *JCP* **2008**, *3*, 16–22. [CrossRef]
34. Bakar, M.Z.A.; Samad, R.; Pebrianti, D.; Mustafa, M.; Abdullah, N.R.H. Finger Application Using K-Curvature Method and Kinect Sensor in Real-Time. In Proceedings of the 2015 International Symposium on Technology Management and Emerging Technologies (ISTMET), Langkawai Island, Malaysia, 25–27 August 2015; pp. 218–222.
35. Nguyen, H.D.; Kim, Y.C.; Kim, S.H.; Na, I.S. A Method for Fingertips Detection Using RGB-D Image and Convolution Neural Network. In Proceedings of the 2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), Guilin, China, 29–31 July 2017; pp. 783–785.
36. Sanchez-Riera, J.; Srinivasan, K.; Hua, K.-L.; Cheng, W.-H.; Hossain, M.A.; Alhamid, M.F. Robust rgb-d hand tracking using deep learning priors. *IEEE Trans. Circuits Syst. Video Technol.* **2017**, *28*, 2289–2301. [CrossRef]
37. Molchanov, P.; Gupta, S.; Kim, K.; Kautz, J. Hand Gesture Recognition with 3D Convolutional Neural Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Boston, MA, USA, 7–12 June 2015; pp. 1–7.
38. Hoang, N.N.; Lee, G.-S.; Kim, S.-H.; Yang, H.-J. A Real-Time Multimodal Hand Gesture Recognition via 3D Convolutional Neural Network and Key Frame Extraction. In Proceedings of the 2018 International Conference on Machine Learning and Machine Intelligence, Hanoi, Vietnam, 28–30 September 2018; pp. 32–37.
39. The 20BN-JESTER Dataset. Available online: https://20bn.com/datasets/jester/v1 (accessed on 10 December 2018).
40. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet Classification with Deep Convolutional Neural Networks. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1097–1105.
41. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
42. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going Deeper with Convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1–9.
43. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 770–778.
44. Baccouche, M.; Mamalet, F.; Wolf, C.; Garcia, C.; Baskurt, A. Sequential Deep Learning for Human Action Recognition. In Proceedings of the International Workshop on Human Behavior Understanding, Amsterdam, The Netherlands, 16 November 2011; pp. 29–39.
45. Schuldt, C.; Laptev, I.; Caputo, B. Recognizing Human Actions: A Local SVM Approach. In Proceedings of the 17th International Conference on Pattern Recognition, Cambridge, UK, 26 August 2004; Volume 3, pp. 32–36.
46. Tran, D.; Bourdev, L.; Fergus, R.; Torresani, L.; Paluri, M. Learning Spatiotemporal Features with 3D Convolutional Networks. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 4489–4497.
47. Shou, Z.; Wang, D.; Chang, S.-F. Temporal Action Localization in Untrimmed Videos via Multi-Stage Cnns. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 1049–1058.

48. Molchanov, P.; Yang, X.; Gupta, S.; Kim, K.; Tyree, S.; Kautz, J. Online Detection and Classification of Dynamic Hand Gestures with Recurrent 3D Convolutional Neural Network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 4207–4215.
49. Pan, Y.; Mei, T.; Yao, T.; Li, H.; Rui, Y. Jointly Modeling Embedding and Translation to Bridge Video and Language. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 4594–4602.
50. Dietterich, T.G. Ensemble Methods in Machine Learning. In Proceedings of the International Workshop on Multiple Classifier Systems, Cagliari, Italy, 21–23 June 2000; pp. 1–15.
51. Gonzalez, R.W.R. *Digital Image Processing*, 3rd ed.; Prentice Hall: Upeer Saddle River, NJ, USA, 2008.
52. Pradhan, R.; Kumar, S.; Agarwal, R.; Pradhan, M.P.; Ghose, M.K. Contour line tracing algorithm for digital topographic maps. *Int. J. Image Process* **2010**, *4*, 156–163.
53. Cassels, J.W.S. *An Introduction to Diophantine Approximation*; Cambridge University Press: Cambridge, UK, 1957; Volume 1957.