Low Cost Hand Gesture Recognition System Design and Implementation

Conference Paper · January 2019

DOI: 10.1109/RoboMech.2019.8704811

3 authors, including:

Nabeel Vandayar
AccioTech
2 PUBLICATIONS **5** CITATIONS
SEE PROFILE

Ken J Nixon
University of the Witwatersrand
52 PUBLICATIONS **86** CITATIONS
SEE PROFILE

LOW COST HAND GESTURE RECOGNITION SYSTEM DESIGN AND IMPLEMENTATION

Nabeel Vandayar*, Timothy James McBride*, Kenneth John Nixon*

*School of Electrical & Information Engineering
University of the Witwatersrand
Johannesburg, South Africa
Email: Nabeel.Vandayar@students.wits.ac.za, Ken.Nixon@wits.ac.za

***Abstract*—The design and development of a low cost hand gesture recognition computer interface, using a standard laptop webcam is presented. The purpose of the system is to recognise both static and dynamic hand gestures from a user in real-time and perform basic macro instructions on the Windows operating system. The calibration and gesture recognition processes are discussed. The system is able to correctly classify 19 static hand gestures and recognise 6 dynamic hand gestures with greater than 95% accuracy. However, the system has a varying latency of between 50-500 milliseconds due to inefficient interfacing with the operating system. It was concluded that an optimised operating system interface would improve system performance and user experience dramatically.**

***Index Terms*—Calibration, Dynamic, Hand gesture recognition, Operating system, Static.**

I. INTRODUCTION

The development of new human-computer interaction (HCI) devices follows the creation of a new type of user interface (UI). The keyboard was created to interface with a textual UI; the mouse and touch screen are used for interacting with a graphical user interface (GUI). The increasing popularity of three dimensional (3D) applications therefore necessitates the creation of a new form of HCI because traditional devices are not designed to operate in 3D space [1] and therefore separates the user from their computing environment [2]. Hand gesture recognition (HGR) is a feasible solution to this problem because gestures are a natural form of human communication [3]. Most attempts to implement HGR require expensive hardware such as depth sensors; high-definition (HD), high-speed cameras and top of the range computers with powerful graphical processing units (GPUs) to process data parallelly so that gestures are recognised in real time.

A low cost, real-time HGR system was created to address this problem by using only a standard laptop web-camera (webcam) as input and only a laptop's CPU for processing data. The GPU is not used because not all laptops have a dedicated GPU and the main constraint for the system is cost. The system interfaces with the computer (Windows operating system) to perform various common tasks such as media control. The system filters out non-gestures to avoid unintentional user inputs. It was assumed that the system is operated by one person at a time, using both hands simultaneously within an environment containing a static background. Success was based on the system being able to recognise a minimum of 10 static and dynamic gestures with an accuracy of greater than 95% and a latency of less than $300\ ms$ while other media applications are running simultaneously.

The designed HGR and HCI systems are presented. The overall performance is discussed and alternative solutions are recommended.

II. BACKGROUND

A. HGR feasibility as a HCI

The development of virtual assistants such as Alexa, Bixby, Cortana and Siri created a new HCI for *smart-device* users. These assistants are HCIs which allow users to issue verbal queries or instructions to the device without having to manually input each command. However, this type of HCI is not efficient when trying to manipulate virtual objects or provide any sort of dynamic input to the computer. A hand gesture interface is more suitable for these tasks because of the many possible gesture combinations which could be applied to virtual objects as they are applied to objects in the real world.

The progressive development of HCI systems such as keyboards, mice and touch screens is a response to a new type of user interface (textual, graphical and screen respectively). As such, each HCI system is designed for a specific type of interface. Although a mouse and touch screen may be able to provide input to a computer, they do not provide mechanical feedback like a regular keyboard. Similarly, a HGR-HCI system may be able to control a cursor but it is uncomfortable for users to maintain a stable pose while controlling the cursor. Cursor control requires an easily recognisable point which could be tracked. A possible solution explored by Xu [4] is to use Kalmin filtering to smoothen the hand position signal after it has been detected. Other solutions require wearable identifiers such as coloured markers which stand out from the background and can be tracked independently[4]. These solutions are ignored because the system would then require additional components to operate, which detracts from the concept of a natural UI.

B. Convolutional Neural Networks

Convolutional neural networks (CNNs) have recently emerged as a means to classify objects in images with great

accuracy [1]. CNNs have been shown to provide accurate image classification results for a wide variation of lighting conditions and object orientations [5]. HGR is a video classification problem because each gesture is made up of a sequence of intermediate gesture poses. The various layers of a CNN are explained in Section V.

A recurrent 3D CNN, developed by Molchanov [6], uses a recurrent layer to *learn* sequences of static gesture poses and label these sequences as corresponding gestures in real-time. Users perceive a delayed response greater than 100 milliseconds after their gesture to be annoying. The system developed by Molchanov [6] detects static and dynamic gestures 560 milliseconds before the pose sequence is complete, therefore the system has a negative lag time. The system has a true positive rate (TPR) of 88% but requires four Titan X GPUs to achieve the aforementioned performance due to the size of the input layer and computational complexity of the recurrent layers [7]. Recurrent layers are therefore good for sequence classification but require far too much computational power to be included in a low cost HGR solution.

Other systems make use of gesture representative frames [7] instead of a video stream input to a CNN. This reduces the size of the input layer and therefore decreases computation time, however preprocessing must be performed to determine which frames are representative of the gesture.

C. MATLAB-Windows Interface

MATLAB is used to develop both the HGR and HCI systems because the availability of various toolboxes allows fast code development time. The Hardware Support Package allows simple access to a laptop webcam. MATLAB also allows a user to execute Microsoft Disk operating system (MS-DOS) commands to perform various functions such as keyboard key presses, adjust speaker volume and screen brightness. Macro instructions are combinations of MS-DOS commands and may be triggered when a specific gesture is performed.

III. HGR-HCI SYSTEM DESIGN

A. User gesture interface

The system is designed for use by an individual on a personal laptop and is easy to set up and use. The gestures to be classified are both simple and intuitive for most people to perform comfortably. Gestures are determined by the position of the hand in the frame but not any explicit finger pose. This constraint may limit the total number of gestures which can be performed but allows individuals without much digital dexterity to be able to perform most of the defined hand gestures. A GUI is implemented only when required for data acquisition, otherwise the program operates in the background.

B. Skin extraction and background subtraction

McBride [8] conducted an investigation of skin extraction, background subtraction and other image processing techniques to aid HGR. Varying lighting conditions and background leakage caused by hand-like structures in the user's environment can not be filtered out with sufficient accuracy. (See McBride [8] for more details.) Therefore these methods are not implemented in the final HGR-HCI system.

C. Personalised CNN model

Although a generalised model is easier to use than a system which requires user calibration, the accuracy of a generalised CNN (trained using many images of various users within different environments) is usually less than that of an over-fitted model (trained using few images of a single user in their current environment). Therefore the calibration phase is designed to fit the model to the user as quick as possible. The model is capable of being recalibrated when relocated to a different environment, while still retaining images from prior calibrations. The system builds a repository of user poses in various environments and will therefore require less calibration over time.

D. System phases

The overall system is composed of two user phases: calibration and implementation as shown in Figure 1 and Figure 2 respectively. Each phase consists of HCI and HGR subsystems (depicted as blue and red boxes respectively).

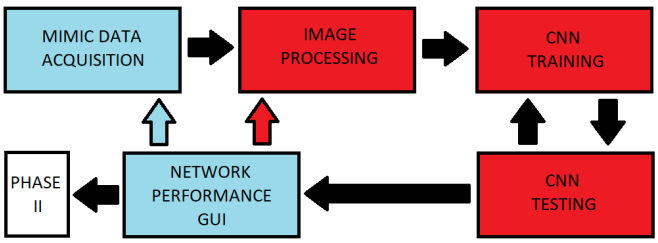

Fig. 1: Calibration Phase

During the calibration phase the user is requested to perform 19 gestures with both arms simultaneously. A model is displayed on screen performing the required gestures for the user to copy. The first pose is of the user in a natural, comfortable position performing no gesture at all. The following nine gestures require the user to position their hand within individual boxes of the grid in a mirrored fashion shown in Figure 3. The final nine gestures again require the user to position each hand in a box corresponding to the model, but at a distance nearer to the screen. This is required for detecting gestures in 3D. The images are then processed as explained in Section V. The CNN is trained and tested until it achieves an accuracy greater than 95%.

The calibration GUI allows the user to save or overwrite new calibrations as needed if the system does not perform accurately enough at any time (depicted by the blue arrow in Figure 1). The user may also retrain the model using existing data (depicted by the red arrow in Figure 1) because the training procedure rarely produces a model of insufficient accuracy. This concept is addressed in more detail in Section V.

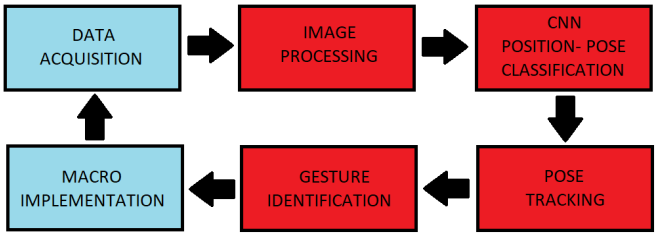

Fig. 2: Implementation Phase

Once the system has been calibrated appropriately, it can be implemented to allow the user to interact with the computer. RGB images are acquired from the laptop webcam and passed the the HGR subsystem which processes the image as explained in Section V. The image is then passed to the CNN which classifies hand position in 3D space using grid coordinates shown in Figure 3. The position of each hand is tracked over time and the sequence of hand positions is identified as one of many complex gestures. The HGR subsystem then passes the gesture label to the HCI subsystem which implements an associated macro instruction before restarting the cycle as shown in Figure 2.

IV. HCI SYSTEM DESIGN

The HCI system takes snapshots of the user via a webcam and passes the image to the HGR system. The HCI then obtains a gesture label from the HGR and carries out an associated macro instruction.

A. Webcam resolution

The webcam is configured to take snapshots at the lowest possible resolution (320×180 pixels during testing) because the HGR system requires a minimum image size of 100×100 pixels to operate. The frame rate of the webcam is dependant on the HGR subsystem as explained in Section VI

B. Gesture grid space

During both the calibration and implementation phases, users are required to perform gestures within the grid space shown in Figure 3. The grid is a virtualisation used to guide users when performing gestures similar to how a keyboard has labels on each of its keys. During the calibration phase users are required to mimic gestures performed by a model. The user must position themselves such that their head is at the mid-line of the image at the intersection of the second line from the top. This position places the user approximately 1.5 arm-lengths away from the laptop screen, which is a comfortable distance for interacting with the device. This limitation is made because the further away a user is from the webcam, the smaller their gestures are and the more difficult it is to distinguish gestures.

The division of the grid in Figure 3 into nine non-overlapping blocks is done using the Golden Ratio. During testing it was observed that when a user elevates their hands to perform a gesture from a distance, their hand is naturally positioned in one of the nine blocks more often than not. The lower regions are the smallest because the hand does not naturally occur in those regions when the user is at the specified distance. The size of the central regions approximates the size of the average hand at those locations. This division was found to reduce the number of false positive classifications made by the HGR as the hand transitions from one block to another because the hand crosses a minimum number of edges as opposed to an evenly spaced grid.

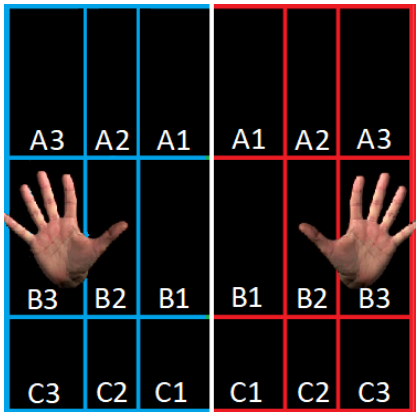

Fig. 3: Golden Ratio Grid

C. Macro implementation

Macro instructions are performed when certain gesture combinations are performed by the user. The user's left hand is used to determine the type of setting they want to adjust or macro to activate. The user's right hand is used to adjust the corresponding setting. Simple static and dynamic gestures are recognised such as swipe and push or pull. For example: the user may raise their left hand to the top-left blue square which indicates that they wish to change the system volume. The user may then raise or lower their right hand to increase or decrease system volume respectively.

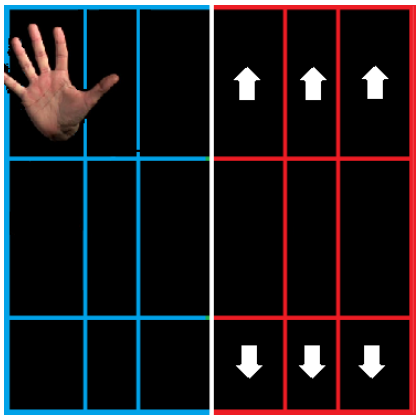

Fig. 4: Volume control gesture.

A list of gesture combinations are included in the Appendix. The purpose of the left-select, right-enforce method is to eliminate unintended gestures from the user and to make most use of the grid space for multiple macro instructions without overly complex gestures for each setting.

V. HGR SYSTEM DESIGN

The HGR system obtains an RGB image from the HCI system, processes it, and provides a gesture label back to the HCI. The HGR process is the most computationally expensive part of the system due to the number of calculations performed to classify a gesture.

A. Image processing

The image is resized to 100×100 pixels to reduce the size of the CNN input layer and therefore the number of operations needed for each classification, which improves overall system performance. The image cannot be scaled down more than this because critical features of the user's hand would be lost.

The image is then split into left and right halves so that each side can be classified separately. The right half is flipped horizontally before being passed to the model for classification as shown in Figure 5. The technique allows the model to be trained with twice the number of training images (left and right sides are both used to classify gestures in corresponding blocks of the grid) and reduces the size of the model by half. This technique explains why users are requested to perform mirrored gestures during the calibration steps. Each half side gesture can be combined with any other half side gesture and effectively increases the total number of possible gestures while reducing the the number of calibration images needed to train the network.

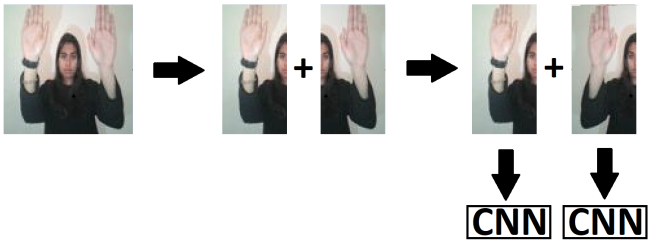

Fig. 5: Image processing procedure.

B. Data Augmentation

During the data acquisition step in the calibration phase of the system, multiple images of the user are taken by the webcam and saved to disk. These images are used to train the model to classify user input during the implementation phase. The CNN should be trained on a variety of similar images because the user is likely to perform gestures in slightly different positions from the those performed during calibration. MATLAB performs data augmentation on both the training and validation data sets automatically. Each image is randomly translated and scaled to create an augmented data set. Data augmentation reduces over-fitting in order to make the model more robust to variation.

C. CNN network design

The CNN is the central component of the system because it classifies a user's hand position in 3D space. The grid in Figure 3 depicts a virtual 2D plane in which hand poses are classified by the coordinates they occur in. A detailed CNN architecture table listing the components of each layer is presented in the Appendix. The CNN has an image input layer of dimensions: $100 \times 50 \times 3$ which accounts for half of the scaled down webcam image and each RGB colour channel. The network has six hidden layers, each composed of a convolution layer, batch normalisation, Rectified Linear Unit (ReLU) activation function and Maximum Pooling components [9].

Each convolution layer consists of a number of filters, each of dimension 3×3 and taking single-step strides with one layer of padding. The number of filters double at each layer from hidden layer 1 (16 filters) to hidden layer 6 (512 filters). This design methodology follows the theme of rich feature extraction at each layer. At the first hidden layer, each of the 16 filters attempt to learn simple features of the input image such as edges and curves. The output of this layer contains richer information than the original image. The number of filters is therefore increased in the next layer to increase the richness of features detected at each subsequent level. This process was repeated until the accuracy of the model was sufficient for HCI purposes but layers were kept to a minimum to reduce computation time and system lag.

Each of the 19 position classifications is represented by 100 augmented images. Each iteration trains the model using 100 images comprised of each positional class evenly. The only exception is the Not-A-Gesture (NAG) class which has double the number of representative images This was done to purposefully skew the classifications towards NAG and avoid position classification unless a gesture is deliberate. The Stochastic gradient descent method is used to minimise the loss function with an L2 regularisation factor of 0.009 and a constant learning rate of 0.01. The default initial weights in MATLAB are determined by a Gaussian distribution with a mean of 0 and a standard deviation of 0.01.

D. Pose tracking and gesture identification

The classified hand positions of each hand must be tracked in time to determine the dynamic or static gesture being performed by the user. A buffer containing the users most recent hand positions is used as a low pass filter to suppress incorrect position classifications from the CNN if they should occur. Only when a certain position fills up the *most-recent-position* buffer, is it added to the next-position buffer. This secondary buffer determines if a user is performing a static or dynamic gesture. If the user performs a static gesture then all components in the *next-position* buffer will be the same. If the user is performing a dynamic gesture then the components in the *next-position* buffer will be different and the associated gesture is determined by geometric rules as follows: If a user's hand changes position from a lower row on the grid to a higher one, then the user must have swiped upwards. The same logic applies in each direction and dimension. The pose tracking algorithm is applied to both hands separately in each dimension. Each hand position or change in each dimension characterises a gesture.

VI. RESULTS

The calibration phase of the system takes a maximum of two minutes to complete. Performing the 19 required gestures needed for training takes one minute but may reduce to 30 seconds as the user becomes more accustomed to the procedure.

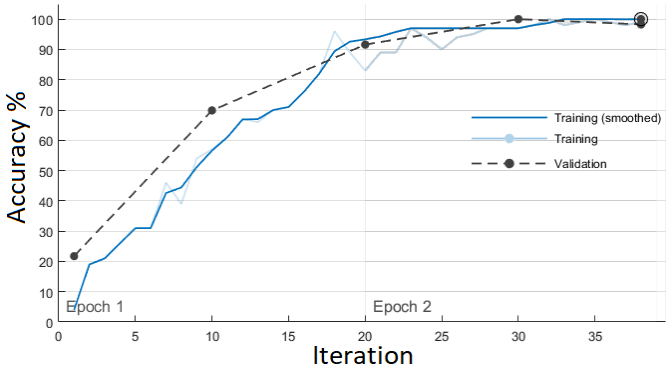

Fig. 6: Accuracy on training and validation sets

Training the CNN takes approximately one minute and usually achieves an accuracy greater than 95% to due the system being over-fitted.

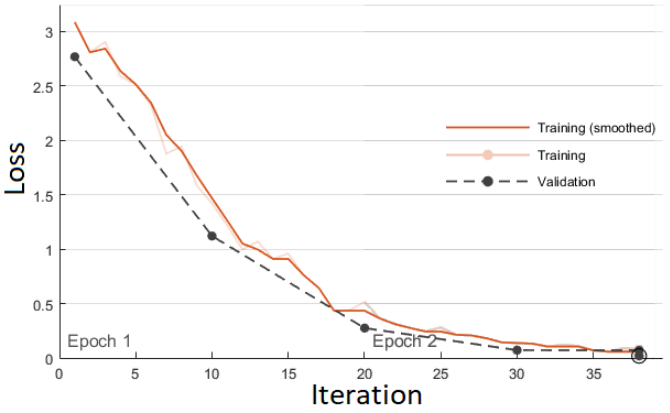

Fig. 7: Loss on training and validation sets

The validation set is evaluated after every ten iterations to reduce training time. After multiple training runs it was discovered that the CNN requires a maximum of 2 epochs (34 iterations each) to train. The loss calculated for the training and validation sets are shown in Figure 7. The fact that loss is better on the validation set than the training set is an indication of over-fitting.

A confusion matrix, shown in Figure 8, is displayed after training the CNN to indicate the classification accuracy of the system for each hand position based on the validation set only. A perfect HGR system should only have values along the diagonal of the confusion matrix because other regions indicate false positive classifications. The orange blocks in Figure 8 indicate that all 15 validation images were classified

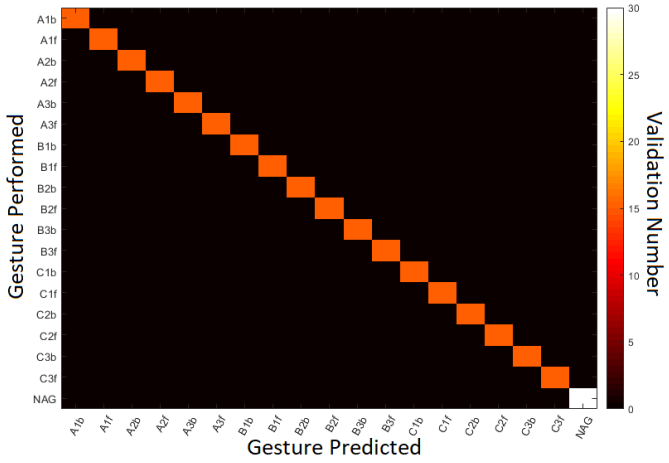

Fig. 8: Confusion Matrix

correctly for each class. The white block indicates that all 30 validation images were classified correctly for the NAG class.

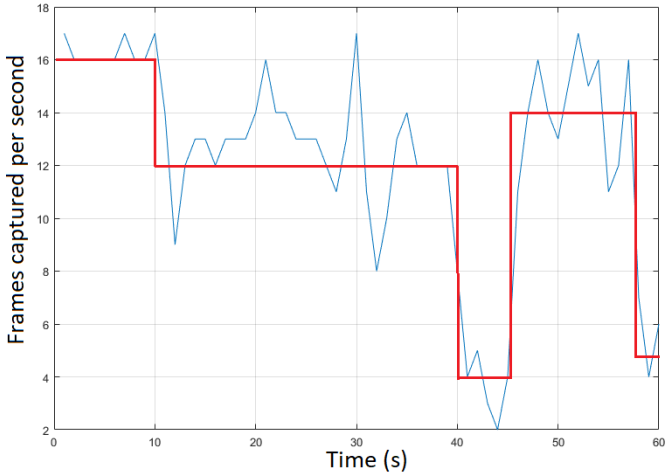

Fig. 9: Frame rate lag when performing gestures

It was observed that the system responds slower when the user performs certain gestures which are mapped to specific macro instructions. An experiment want set up to determine the effect of different gesture-macro combinations on the system response time. The results are shown in Figure 9. The number of frames per second (FPS) captured when no gesture is being performed (first 10 seconds) is 16. The FPS decreases to 12 when the user attempts to change the screen brightness (10-40 seconds). The FPS drop down to 4 when the user attempts to change the volume (40-45 seconds), up to 14 FPS when scrolling and down to 5 FPS when using the key combinations. The system latency therefore varies between 50-500 milliseconds A minimum of 10 FPS (100 millisecond lag) is required for the system to operate in real-time, therefore the lag becomes noticeable when certain gestures are performed.

This is due to a bottle neck caused by repeatedly opening a new MS-DOS command window for each macro instruction instead of having it stay open as long as the system is running.

VII. DISCUSSION

The HGR-HCI system was tested by using the system to control YouTube video playback in the FireFox browser; Google Earth in the Chrome browser and PDF documents in Adobe PDF Viewer as well as other computer settings such as speaker volume and screen brightness. The system was calibrated trained in an acceptable time of 1 minute and 32 seconds. The system was able to classify gestures accurately however system lag affects user experience occasionally.

The system was able to pause, play and scrub a YouTube video without noticeable lag. When controlling the Google Earth globe, the system response was too sensitive and would often overshoot. Unlike the click of a keyboard or mouse, the HGR-HCI system provides no tactile feedback to the user to indicate that a gesture has been detected. The user must rely on auditory or visual cues instead. This is a deficiency of the system because a user is more likely to perform a gesture longer than is required and have to make corrections when controlling an application such as Google Earth. When scrolling through a PDF document, the system was also too sensitive. A sensitivity variable should be added to the system which the user can set without affecting the system response time.

The left-select, right-enforce method of performing gestures prevented unintentional gestures by the user being detected by the system, however it requires the user to hold up their left hand as long as they wish to perform a gesture with their right hand. This is inefficient and leads to user fatigue. An additional limitation is that each hand is only guaranteed to be detected in its half of the grid which is not intuitive to a new user. A different method should be developed for determining when to detect user gestures and what macro instruction to execute depending on context. The system could be integrated with existing virtual assistants to recognise both hand gestures and voice input for a more natural user interface.

The use of the Golden Ratio grid to guide user input makes performing gestures within a confined area more natural. However, resizing the input image from a rectangle to a square, distorts the virtual grid divisions slightly by shrinking the horizontal gesture area. The input layer should therefore be altered to accept rectangular inputs. An alternate solution is to use a shallow regional CNN to identify and crop just the hand itself from an input image and feed the that to a deeper CNN for pose classification.

VIII. CONCLUSION

A low cost hand gesture recognition computer interface was designed and built using a webcam, convolutional neural network and MS-DOS interface in MATLAB. The system is calibrated to an individual user within two minutes and is able to classify 19 static hand gestures and recognise 6 dynamic hand gestures with greater than 95% accuracy because it is over-fitted to the user by design. The system latency varies between 50-500 milliseconds due to the lag caused by the interface with the operating system. This is not sufficient for a real-time user interface. It is recommended that the operating system interface be optimised to remove the bottleneck.

IX. ACKNOWLEDGEMENT

The author would like to thank Professor K Nixon from the School of Electrical and Information Engineering for his guidance, and Dr R Klein from the School of Computer Science and Applied Mathematics for his advice.

REFERENCES

[1] A. Birdal and R. Hassanpour. "Region based hand gesture recognition." 2008.

[2] H.-S. Yeo, B.-G. Lee, and H. Lim. "Hand tracking and gesture recognition system for human-computer interaction using low-cost hardware." *Multimedia Tools and Applications*, vol. 74, no. 8, pp. 2687–2715, 2015.

[3] P. Narayana, J. R. Beveridge, and B. A. Draper. "Gesture Recognition: Focus on the Hands." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5235–5244. 2018.

[4] P. Xu. "A Real-time Hand Gesture Recognition and Human-Computer Interaction System." *arXiv preprint arXiv:1704.07296*, 2017.

[5] P. Molchanov, S. Gupta, K. Kim, and J. Kautz. "Hand gesture recognition with 3D convolutional neural networks." In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 1–7. 2015.

[6] P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree, and J. Kautz. "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*, pp. 4207–4215. 2016.

[7] V. John, A. Boyali, S. Mita, M. Imanishi, and N. Sanma. "Deep learning-based fast hand gesture recognition using representative frames." In *Digital Image Computing: Techniques and Applications (DICTA), 2016 International Conference on*, pp. 1–8. IEEE, 2016.

[8] M. T. "Low cost hand gesture recognition system: Image processing considerations." 2018.

[9] A. Ng. "Machine Learning Yearning.", 2017.

[10] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. "Deconvolutional networks." 2010.

[11] M. D. Zeiler and R. Fergus. "Visualizing and understanding convolutional networks." In *European conference on computer vision*, pp. 818–833. Springer, 2014.

[12] C. Zimmermann and T. Brox. "Learning to estimate 3d hand pose from single rgb images." In *International Conference on Computer Vision*, vol. 1, p. 3. 2017.

APPENDIX

The components of the CNN used in the HGR-HCI system are given in Table I. The CNN has an image input layer, six hidden layers and an output layer. The network locates a user's static hand gesture in 3D space.

TABLE I: Layers of the CNN used in the HGR-HCI system

Layer	**Component**	**Details**
Input	Image Input	100x50x3 images with 'zerocenter' normalization
1	Convolution	16 3x3x3 convolutions with stride [1 1] and padding [1 1 1 1]
	Batch Normalization	Batch normalization with 16 channels
	ReLU	ReLU
	Max Pooling	2x2 max pooling with stride [2 2] and padding [0 0 0 0]
2	Convolution	32 3x3x16 convolutions with stride [1 1] and padding [1 1 1 1]
	Batch Normalization	Batch normalization with 32 channels
	ReLU	ReLU
	Max Pooling	2x2 max pooling with stride [2 2] and padding [0 0 0 0]
3	Convolution	64 3x3x32 convolutions with stride [1 1] and padding [1 1 1 1]
	Batch Normalization	Batch normalization with 64 channels
	ReLU	ReLU
	Max Pooling	2x2 max pooling with stride [2 2] and padding [0 0 0 0]
4	Convolution	128 3x3x64 convolutions with stride [1 1] and padding [1 1 1 1]
	Batch Normalization	Batch normalization with 128 channels
	ReLU	ReLU
	Max Pooling	2x2 max pooling with stride [2 2] and padding [0 0 0 0]
5	Convolution	256 3x3x128 convolutions with stride [1 1] and padding [1 1 1 1]
	Batch Normalization	Batch normalization with 256 channels
	ReLU	ReLU
	Max Pooling	2x2 max pooling with stride [2 2] and padding [0 0 0 0]
6	Convolution	512 3x3x128 convolutions with stride [1 1] and padding [1 1 1 1]
	Batch Normalization	Batch normalization with 256 channels
	ReLU	ReLU
	Max Pooling	2x2 max pooling with stride [2 2] and padding [0 0 0 0]
Output	Fully Connected	19 fully connected layer
	Softmax	softmax
	Classification Output	crossentropyex with 'A1b' and 18 other classes

The following figures show the various gesture combinations used to control different functions of the Windows operating system via the HGR-HCI system. The blue half of each image indicates the position of the user's left hand. The symbols in the red half of each image indicate the gesture the user's right hand must be perform in to initiate the associated adjustment.

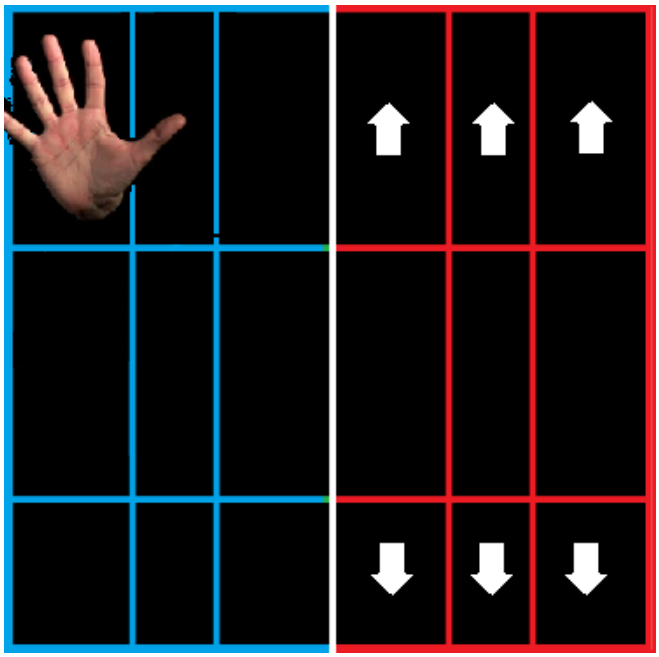

Fig. 10: Volume control gesture
Swipe up or down for volume adjustment.

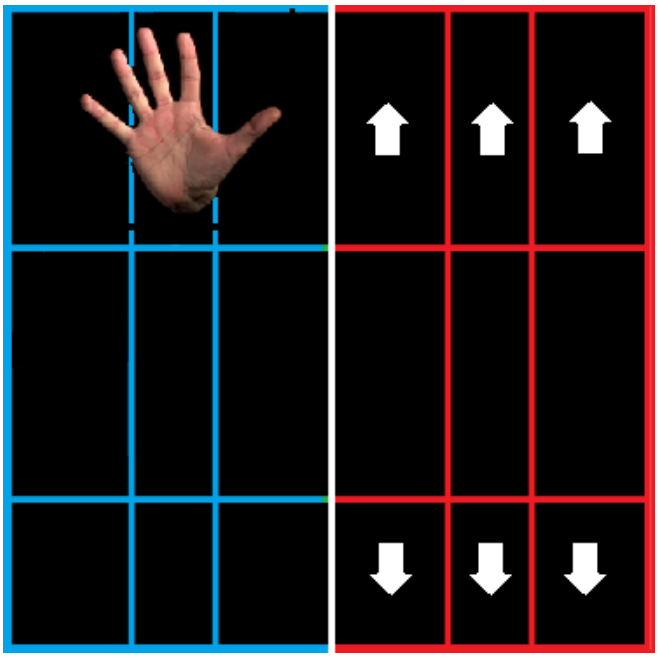

Fig. 11: Brightness control gesture
Swipe up or down to adjust screen brightness.

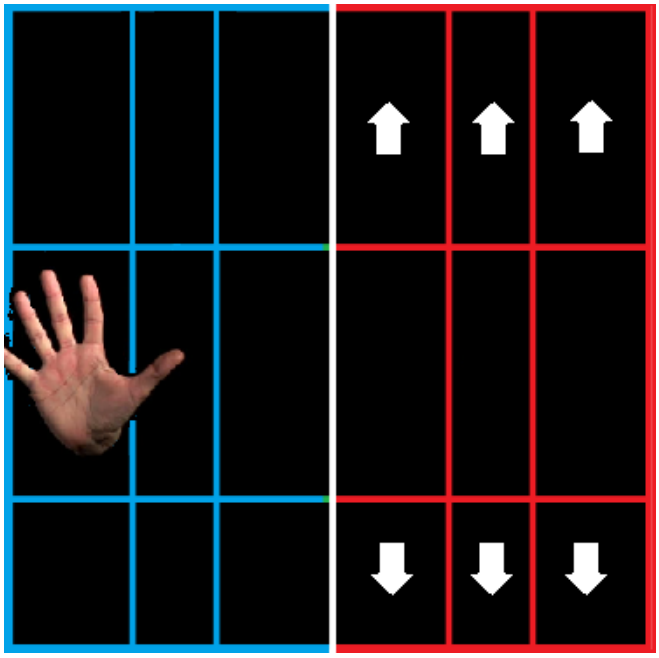

Fig. 12: Scroll control gesture
Swipe up or down to adjust scroll.

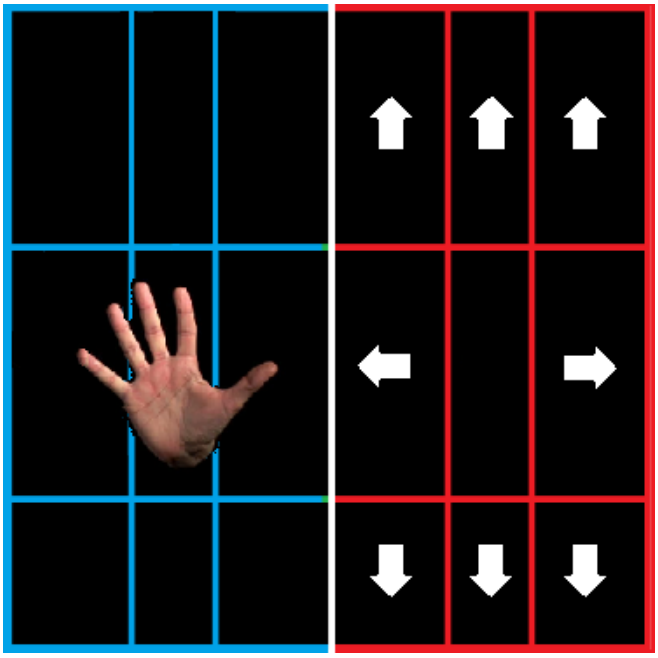

Fig. 13: Arrow control gesture
Swipe up, down, right or left to press arrow keys.

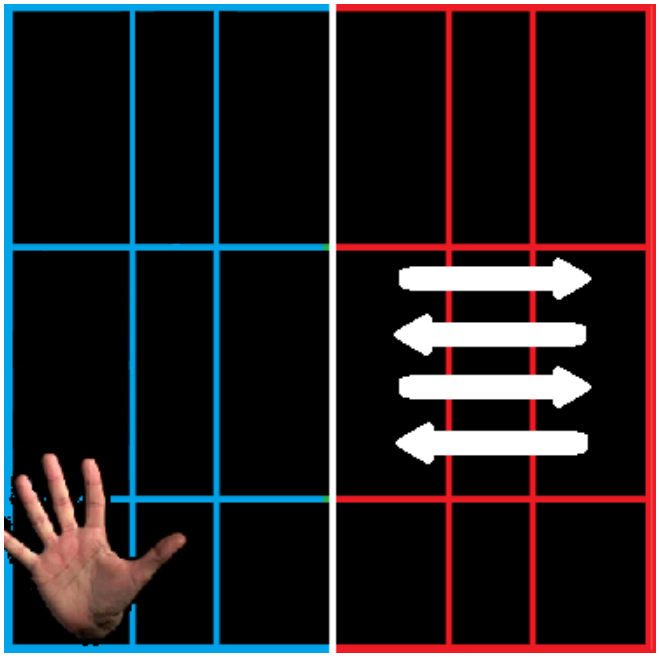

Fig. 14: Bye control gesture
Swipe right, left, right and then left to turn off monitor.

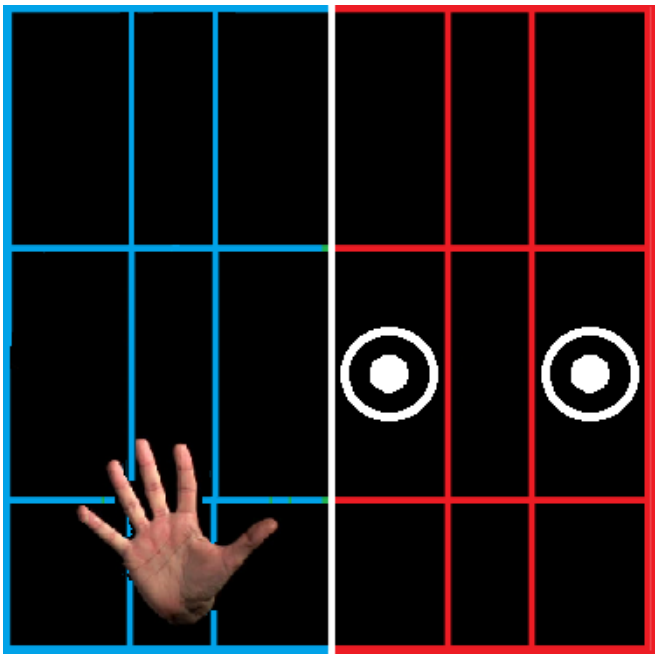

Fig. 15: Enter and ALT+TAB key control gesture
Push hand forward on the left to press the ENTER key.
Push hand forward on the right to press the ALT+TAB key combination.

The associated macro instructions are only performed while both hands are up at the same time. This is to protect against unintentional gestures by the user.