A Hand Gesture Recognition Framework as a Configurable Input Device and for Evaluating Recognition Strategies

Samuel Wright

April 8, 2013

Abstract

After reviewing existing applications and implementations of hand gesture recognition, we propose creating a software package able to learn and act upon user-configured hand gestures from a standard webcam for simple interactions when using a mouse, keyboard, or touchpad is not practical. We then lay out clear goals for what we hope to achieve, with an emphasis on creating a working prototype as quickly as possible, where improvements are easily made and tested through the implementation of a robust and hot-swappable framework that could be useful for academics looking to compare component algorithms of gesture recognition.

1 Background

1.1 Human Computer Interaction (HCI)

HCI is a rich and broad field due to the innumerable tasks a computer can perform and the variety of information required by and from the user for each. Concepts such as direct manipulation of graphical objects and gesture recognition have been heavily researched since the 1960s [1], and have led to the development and general adoption of mice, touchpads, and touchscreens.

There are numerous use cases where such devices are unnatural or inconvenient, such as interacting with a 3D virtual environment or interpreting sign language [2]. Interpreting hand gestures can provide a more natural interface [2], and a data-rich input given that humans have fine control of the 27 degrees of freedom each hand offers [3]. As such, hand gesture recognition has become a prominent and exciting subfield of HCI.

1.2 Applications of Hand Gesture Recognition.

To understand what is meant by a hand gesture in the field of HCI, we now examine the existing applications of hand gesture recognition (from [4]).

1.2.1 Virtual Reality

Hand gestures are used as a means to manipulate 3D objects in software as if the objects were real (for example, transforming complex molecules as a means to better understand their properties [5]). Online, dynamic, 3D gestures are required in order to manipulate the objects (eg. rotation, translation, and scaling).

1.2.2 Robotics and telepresence

Hand gestures are used to control remote objects through a computerised intermediary (for example, a surgeon is able to control a laparoscopic camera via a computer using hand gestures to allow for fine control [6]). The gestures often used are similar to those used in virtual reality applications.

1.2.3 Desktop and tablet applications

Hand gestures are used to manipulate standard PC applications without the use of a mouse (for example, surgeons are able to directly control where to zoom in on images taken during an endoscopy without the need to operate a mouse or relay instructions via an assistant [7]). All manner of gestures (online or offline, dynamic or static, 2D or 3D) could find compelling uses, depending on the specific application and task at hand.

1.2.4 Gaming

Hand gestures are used to control in-game characters or issue commands. The controller for the Nintendo Wii contains a triaxial accelerometer, allowing for in-game control of hand-held objects such as swords. The Xbox Kinect contains a set of cameras that can track hand motions in 3D, allowing for direct interaction

with the in-game environment. As with desktop applications, the type of gestures required depend on the specifics of the game.

1.2.5 Sign language

Hand gestures (alongside arm and face gestures) represent a language that can be interpreted as commands or translated to a written or spoken language. The use of computers can, for example, help people learn sign language by interpreting and verifying their actions (for example, as part of a game [8]). The gestures are complex, 3D and dynamic.

1.3 General User Requirements

For a gesture recognition system to be successful it must be able to recognise enough gestures to sufficiently serve a compelling purpose, however this functionality will only be apparent if the system is adequately usable [9]. This means it must be robust enough to reliably recognise gestures regardless of the complexity of the background and lighting conditions, operate in real-time with a minimum of lag, and be tolerant of user error [4]. The matching of gestures with their functionality must also be obvious to allow for easy learning and adoption of the system [10].

1.4 Existing Consumer Systems

Here is a summary of current hand gesture recognition systems that use only a 2D camera.

1.4.1 Toshiba AirSwing

A commercial hand gesture recognition system to be marketed to digital signage companies. Videos of it in use suggest it can robustly process a predetermined set of offline and online gestures. To augment the lack of a mouse, the screen shows a translucent image of the user so the user's hand represents the cursor.

1.4.2 HandVu

An open source, OpenCV-based hand gesture recognition system that can recognise 6 predetermined static offline gestures, but requires manual initialisation and the hand tracking is often inaccurate.

1.4.3 FlutterApp

A commercial hand gesture recognition system which recognises three predetermined static offline gestures in order to control a variety of media applications. It is responsive and performs well.

1.5 Motivation

With the rise of capacitive touchpads, multitouch gestures are becoming more popular. There are a number of software packages (eg. BetterTouchTool [11]) which allow the user to map multitouch gestures to actions like window management (eg. maximise the current window), control media, or simulate keyboard button presses. The benefit of such a system is that each user can develop their own use cases and workflows. To our knowledge, there is no such equivalent software for hand gestures detected using a webcam.

This is likely due to the complexity of hand gesture recognition. There are many elements to a gesture recognition system (as described in the next section) and often the best choice of elements is dependent on the use case. There currently does not exist a framework for evaluating and comparing different strategies - even academics whom publish their ideas about new strategies will only compare against a handful of others in single-use scripts.

Our proposal is two-fold. First, we intend to develop a gesture recognition framework that will allow different strategies to be tested against each other. We will record a variety of images of hand gestures in different circumstances (for example, with differing background complexities) so testing a strategy will reveal its objective weaknesses and strengths. In order to understand these, the framework will allow for debug-like tools, such as being able to pause recognition and inspect the state of the image at each point in the strategy. This tool could be useful for academics to easily test and optimise their new ideas.

Secondly, we intend to develop the BetterTouchTool for hand gestures, on top of the framework. It should be able to associate hand gestures with offline actions such as simulating a keyboard button press, and also with online actions such as changing the speaker volume by raising or lowering the hand.

2 Relevant Work

We are interested in the specifics of the various implementations of hand gesture recognition. The general workflow employed is data capture, feature extraction, then gesture classification.

2.1 Data Capture

There are two categories of sensors which are to capture hand gestures: hand-mounted and vision-based [4] which we now discuss.

2.1.1 Hand-mounted sensors

Hand-mounted sensors (for example those found on a "data glove" [12]) provide each digit's joint orientations (either through mechanical or optical sensors) in real-time. While accurate and reliable, these require the user to wear bulky equipment which inhibits their adoption for general use. They have proven useful in learning sign languages [8], where the gestures are complex enough for the glove's reliability and accuracy to be useful, and are used for long-enough periods of time to make them worthwhile.

2.1.2 Vision-based sensors

This approach does not require the user to wear any sensors, allowing for a more natural and convenient input of data (although not entirely natural, since the gestures must be made within the camera's field of view). Hands have proven to be notably difficult to detect and interpret from visual data [13] because they are homogenous in colour and texture, lack static features (compared to a face, for example) and have many degrees of freedom. Depth-sensing [14] or IR cameras [15] can be used to separate the hand from other objects while providing extra data which can be used to increase the fidelity of the features extracted. Such cameras are still expensive and are not as widely available as 2D cameras.

2.2 Feature Extraction

If the input device is a data glove, the data it captures (ie. joint orientations for all digits) can be immediately used as features. If instead a camera is used, the data it captures requires substantial processing, typically involving hand detection, pose estimation and hand tracking [16]. The correct method of feature acquisition to use in a gesture recognition system is a function of its desired application, the environment in which it will be used, and the notion of what is reasonable to ask of the user in terms of money and preparation time.

2.2.1 Hand detection

Detecting a hand from a 2D camera presents a number of difficulties. The hand's silhouette is complicated and dynamic, and its colour is dependent on the specific user's skin colour and the lighting conditions. A hand's colour is relatively uniform throughout gesturing (as opposed to its edges), so is a good and computationally-cheap initial step in hand detection.

Skin colours and non-skin colours can be separated using classifiers working in the hue-saturation-intensity (HSI) colour-space [17] and variations in detected colour due to lighting can be circumvented using colour-space statistics [18]. Face-detection (which is easily done due to the face's static shape) can be used to track the subject's skin colour more accurately [19]. More advanced learning-based techniques also exist, for example using a self-organising map to reduce the number of dimensions of an image's colour space as a means of classifying skin-coloured objects [20].

Once the skin-coloured areas of an image have been identified, one could simply mask the rest of the image. However, this might introduce noise into the hand's subimage if there were abnormally dark shadows (ie. a false negative) or it might find small skin-coloured areas outside of the hand (ie. a false positive). An alternative approach is to create a image where the intensity of each pixel is the scaled probability of the corresponding pixel from the camera being skin-coloured, then using an algorithm such as MSER [18] which attempts to find large regions of interest (in this case, large skin-coloured objects). Many algorithms exist which find regions of interest, though MSER has been shown to be one of the most effective [21].

Detecting skin colours will also pick up faces and other body parts, so more discrimination is required for such complex backgrounds. A workaround would be to require the user to make a pre-determined hand shape (eg. an open palm facing the camera) before performing their gesture, which could be detected by fitting a simplistic model of the hand's shape to the image [22]. Such a workaround adds inconvenience to using a gesture recognition system, so should be avoided if possible. Facial recognition can again be used to avoid confusion with a hand, even when the hand partially occludes the face [23]. Smith et al [24] have proposed an interesting method that uses an analogue from physics (force and energy fields) to detect regions of complexity in an image such as a hand.

An alternative and popular [25, 26, 27, 28] method decomposes the image (which can have a complex background) into Haar-like features (which describe local image-intensity distributions across the image) before being passed to a set of weak classifiers that have been trained on a large set of hand images, and together form a strong classifier for detecting the presence of a hand. The AdaBoost algorithm [29], which improves the performance of the classifier, is often used. This approach has shown promise with small vocabularies [26] but training large vocabularies is computationally expensive [30]. The algorithm used to generate Haar-like features [25] relies on using integral images [31] which are scale invariant, so generating the Haar-like features on different scales does not require recalculating the integral images, making this a very

efficient technique. Classifying the features is done in a "cascade" [25] of weak classifiers, which allows for easy parallelisation of the fail-fast classifiers, so very little time is wasted processing parts of the image that don't contain a hand.

2.2.2 Pose estimation

After the hand has been detected, a feature set must be extracted from it that will be used to recognise the gesture being made. There are two approaches [3], model-based and appearance-based, which we now discuss in turn.

Using a 3D computerised model of a hand which faithfully mimics the available ranges of motion (ie. a kinematic model) a variety of poses and viewpoints can be tried to find the 2D projection that matches the image of the hand from the sensor [32]. Any number of metrics can then be taken from the model hand and used as features for gesture recognition. This provides as much data as the data glove, but requires the edges of the hand to be accurately recorded (which can be difficult if the background does not contrast with the hand, or the hand self-occludes) otherwise the computerised model can lose much of its fidelity [4]. Another issue is the complexity of finding the 3D model that produces the measured silhouette (an example of inverse kinematics), which is a non-linear problem due to the trigonometric functions required [30]. If subsequent images in the video show only small differences in the hand's pose, these functions can be approximated linearly, but otherwise can require non-trivial amounts of computational time. The additional use of depth-sensing cameras [13] lifts these restrictions and improves the reliability of the metrics generated, although depth-sensing cameras are still non-standard and expensive compared to RGB cameras.

Alternatively, the appearance of the hand as the camera sees it can be used without inferring knowledge about the state of the hand. For example, Gupta et al. [33] use the EigenTracking algorithm [34] to generate an eigenspace representation of the hand which is invariant under affine transformation. This effectively condenses a large sequence of *a priori* hand images into a small set of images which represent hand-like qualities, and form the basis vectors of the eigenspace. The image to be tested is then decomposed into a linear sum of these basis vectors, resulting in the image's coordinates in the eigenspace which can then be used as a feature set describing the hand shape (regardless of how it is rotates or translates). However, the compactness of the eigenspace (which is required for an efficient description of the hand state) depends on the gesture vocabulary being small. [30].

Another appearance-based approach is to generate Haar-like features (as mentioned in section 2.2.1). Instead of detecting whether there is or is not a hand, the features would be used to classify the hand pose, thus requiring even more training than was required for hand detection.

2.2.3 Hand tracking

The hand-detection algorithms that automatically initialise can be used in every frame to keep track of the hand, but that would ignore the coupling between image frames in a video. This would be good if the framerate is too low for the speed at which the hand is moving, but otherwise it ignores a wealth of data that could be used to track the hand. By using a separate hand tracking algorithm which uses this data, errors in either algorithm can be smoothed out.

A popular object-tracking technique is the CONDENSATION algorithm [35] which is capable of tracking curves against complex backgrounds, as in [33], and has been shown to be computationally efficient. Another technique is the Camshift algorithm [36] which has been shown to perform well with noise, distractors (eg. other hands), and partial occlusion [37].

2.3 Classification

Classification compares the features extracted from a gesture against those from a vocabulary of gestures, selecting the most similar. In this proposal we are only concerned with classifying static gestures (ie. classifying the hand pose in each frame separately).

The features on which to classify a hand might describe concrete data about the hand, such as digit joint angles or an image of the hand's silhouette, or they might describe more abstract data such as Haar-like features of the image. The best features to use will be invariant to translation, rotation, scaling, and changes in lighting (or at least robust to such changes), though a lack of invariance can be overcome by a more complex classifier and a more diverse training set [27].

The correct classifier to use, and its morphology, must be determined empirically by monitoring performance and computational efficiency. The AdaBoosted cascade of weak classifiers, as described in section 2.2.1, has shown to be a good classifier for high-dimensional features [25]. Another popular classifier [38, 39] is the support vector machine (SVM), which is a statistical approach to segregating the feature space. Others [40, 41] have used artificial neural networks, which comprise highly-connected processing units inspired by neurones that are trained iteratively through error-minimisation.

3 Implementation

3.1 Key Goals

3.1.1 Implement a simplistic recognition strategy

First, we will create a simple strategy to be able to recognise static gestures against a white background. The hand detector will mask anything close to white. The pose estimator will fit a circle to the palm and use edge detection outside of the circle to find fingertips, then use the distance from the fingertips to the circle as features. Since the hand is static, no hand tracking will be required. Classification will be choosing the vocabulary image with finger lengths that are closest to the test image.

Each part of the gesture recognition strategy (ie. hand detection and tracking, pose estimation and classification) will be represented by an element object, and will belong to a strategy object that will perform recognition by passing images between the elements inside mediator objects. These images will be saved or displayed to demonstrate the strategy's performance.

3.1.2 Develop the framework's test functionality

Next we will create a GUI that allows for strategies to be created with customisable elements then executed, with an image displayed after each element has finished (except for after classification, where the class' name should be displayed). The strategies' configurations should then be able to be saved to disk.

We will then create a GUI to allow for sets of hand images and videos to be taken from either a camera or a file, and for each item to be tagged with a gesture name and any other information (eg. how complex the background is, whether the hand is rotated or self-occluding). We can then populate the system with a variety of gestures of differing complexity.

Once done, we will create another GUI that pops up after a test has been run that displays general performance as well as performances indexed by image tag (eg. this strategy had a 30% success rate with images tagged "complex background") and the amount of CPU time used per frame.

3.1.3 Implement the hand detection and pose estimation elements

We will implement the algorithms described in section 2.2.1 and section 2.2.2. Most of them already have open source implementations (in OpenCV) that would only need wrapping. For now, hand tracking will be done by the hand detection algorithms.

3.1.4 Improve the test framework

By this point, the elements are complex enough to need configuration panels. Using something like the "jfg" package, we will create a configuration form given each element's requirements that is displayed in the test GUI when an element is clicked. The configurations should be able to be saved to disk as part of each strategy.

We will add more data to the mediator object, for example the hand detector could store the height and width of the bounding box containing the hand, which can be displayed by hovering over the corresponding output image.

3.1.5 Implement hand tracking algorithms

Both the CONDENSATION and camshift algorithms are implemented in OpenCV, so we only need to wrap them in the element class. In the image in the test GUI that represents the output of the hand tracking algorithm, we will show the output of the hand detection algorithm in another colour (for comparison with the hand detection routines).

3.1.6 lmplement simple action configuration GUI

We will create a GUI that allows a user to link hand gestures to simulated keyboard button presses. Actually simulating the button presses is simple using the "java.awt.Robot" package, and since there is no further action required after the button is simulated this represents a good first step.

The class that simulates the button press shall be an "action handler". It does not know anything about the underlying framework except the mediator object, which contains the gesture class and other information that more advanced action handlers might find useful.

The GUI should allow for each action handler to be configured, using the same technique as was used in configuring the gesture recognition elements. The keyboard button press simulator would need to know which key combination to simulate, for example.

If a gesture is wrongly classified, the user should be able to choose the correct class from a list and the images that were wrongly recognised should be added to the training set for the correct class.

3.2 Aspirational Goals

3.2.1 Allow the user to record static gestures

This would use the same GUI as used in section 3.1.2 to take images from the camera of the gesture from slightly different angles (to form a valid training set),

then retrain the classifiers. If the classifier doesn't perform well, the gesture was probably too similar to a previously defined gesture, which the user should be told.

3.2.2 Allow for dynamic strategies

We will create strategy elements that dynamically delegate to other elements based on previous mediator objects. For example if the background is complex (which could be calculated every few minutes) then choose a hand detector that can cope with it. The rules for changing element would be hard-coded into the delegator elements, but could be modified using the configuration forms described in section 3.1.4.

3.2.3 Implement a cross-platform window resizing and moving package

We will create a package to close and minimise windows, and resize them to the left or right half, the top or bottom half, or to any of the quadrants of the screen.

On Mac, windows can be managed via AppleScript, which can be executed from within java using the "javax.script" package. On Windows there are "win32" handles to manage windows, which can be accessed using the "JNA" java package. On Linux, windows can be controlled from the shell utility "wmctrl", which can be called using the "Runtime" java class. The action handler to use this package would choose which method to use at runtime based on the OS. Its configuration panel could be a drop-down menu of the aforementioned functions.

3.2.4 Implement simple non-static gestures

These are gesture where the hand pose determines the class (and thus the action to take) and subsequent translation and rotation is passed to the action handler for it to decide what to do. The action handler would receive mediator objects for every frame after the pose was detected (which will contain hand translation and rotation since the last frame). When the mediator object reports that the hand has changed its class, the action handler can decide to release control back to the framework to choose a new action handler.

For example, an action handler could manipulate the speaker volume by tracking the hand's height in the image. Java can manipulate speaker volume using the "javax.sound.sampled.AudioSystem" package. The action handler's configuration panel would control the sensitivity and direction of hand motion to use.

3.2.5 Implement dynamic gestures

By dynamic, we mean gestures which includes multiple hand poses. This would be handled by the action handler when it is given a mediator object that shows the hand's pose class has changed. Instead of immediately returning control back to the framework however, it can choose a new action.

For example while changing the volume the user might extend a finger, which the volume changing action handler could interpret as a desire to mute the speakers.

3.3 Expectations

We expect to complete at least up to section 3.1.5 by which point we will have implemented a useful test framework for hand recognition. If the recognition is not accurate (eg. static gestures in front of complex backgrounds are correctly classified less than 80% of the time) then there would be little point in creating the action handlers. Instead we would continue to improve the test framework by completing section 3.2.2, and investigate optimising the implemented methods or try other methods available in OpenCV, or look at recent articles and implement promising algorithms. Otherwise we hope to implement as many of the optional goals as time allows.

References

[1] BA Myers. A Brief History of Human-Computer Interaction Technology. *interactions*, pages 44–54, 1998.

[2] Ying Wu and TS Huang. Vision-based gesture recognition: A review. *Urbana*, 51:103–115, 1999.

[3] Ying Wu and TS Huang. For Vision-Based Human Computer Interaction. *IEEE Signal Processing Magazine*, (May):51–60, 2001.

[4] GRS Murthy and RS Jadon. A review of vision based hand gestures recognition. *International Journal of Information Technology and Knowledge Management*, 2(2):405–410, 2009.

[5] R Sharma and TS Huang. Speech/Gesture Interface to a Visual Computing Environment for Molecular Biologists. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference*, pages 964–968. 1996.

[6] G H Ballantyne. Robotic surgery, telerobotic surgery, telepresence, and telementoring. Review of early clinical results. *Surgical endoscopy*, 16(10):1389–402, October 2002.

[7] C Grätzel, T Fong, S Grange, and C Baur. A non-contact mouse for surgeon-computer interaction. *Technology and health care : official journal of the European Society for Engineering and Medicine*, 12(3):245–57, January 2004.

[8] Zahoor Zafrulla, Helene Brashear, Pei Yin, Peter Presti, Thad Starner, and Harley Hamilton. American Sign Language Phrase Verification in an Educational Game for Deaf Children. *2010 20th International Conference on Pattern Recognition*, pages 3846–3849, August 2010.

[9] Fakhreddine Karray, Milad Alemzadeh, JA Saleh, and MN Arab. Human-computer interaction: Overview on state of the art. *International Journal On Smart Sensing and Intelligent Systems*, 1(1):137–159, 2008.

[10] Ping Zhang and Dennis F Galletta. *Human Computer Interaction And Management Information Systems: Foundations*. ME Sharpe Incorporated, 2006.

[11] Andreas Hegenberg. BetterTouchTool <www.boastr.de>.

[12] Thomas G Zimmerman, Jaron Lanier, Chuck Blanchard, Steve Bryson, and Young Harvill. A hand gesture interface device. *SIGCHI Bull.*, 17(SI):189–192, May 1986.

[13] Michael Van Den Bergh and L Van Gool. Combining RGB and ToF cameras for real-time 3D hand gesture interaction. *Proceedings of the 2011 IEEE Workshop on Applications of Computer Vision*, pages 66–72, 2011.

[14] K. Fujimura. Hand gesture recognition using depth data. *Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. Proceedings.*, pages 529–534, 2004.

[15] Y. Sato, Y. Kobayashi, and H. Koike. Fast tracking of hands and fingertips in infrared images for augmented desk interface. *Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580)*, pages 462–467, 2000.

[16] Georg Hackenberg, R McCall, and W Broll. Lightweight palm and finger tracking for real-time 3D gesture control. In *Virtual Reality Conference (VR), 2011 IEEE*, number March 2010, pages 19–26. 2011.

[17] MJ Jones and JM Rehg. Statistical color models with application to skin detection. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference*, pages 274–280. 1999.

[18] Michael Donoser and Horst Bischof. Real time appearance based hand tracking. *2008 19th International Conference on Pattern Recognition*, pages 1–4, December 2008.

[19] J Fritsch and S Lang. Improving adaptive skin color segmentation by incorporating results from face detection. *Robot and Human Interactive Communication, 2002. Proceedings. 11th IEEE International Workshop*, (September), 2002.

[20] Ying Wu, Qiong Liu, and TS Huang. An Adaptive Self-Organizing Color Segmentation Algorithm with Application to Robust Real-time Human Hand Localization. In *Proc. of Asian Conference on Computer Vision*, pages 1106—-1111. 2000.

[21] K. Mikolajczyk, T. Tuytelaars, C. Schmid, a. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A Comparison of Affine Region Detectors. *International Journal of Computer Vision*, 65(1-2):43–72, October 2005.

[22] Taehee Lee and Tobias Hollerer. Handy AR: Markerless Inspection of Augmented Reality Objects Using Fingertip Tracking. *2007 11th IEEE International Symposium on Wearable Computers*, pages 1–8, October 2007.

[23] Matilde Gonzalez, Christophe Collet, and R Dubot. Head tracking and hand segmentation during hand over face occlusion in sign language. *Trends and Topics in Computer Vision*, pages 234–243, 2012.

[24] Paul Smith, Niels da Vitoria Lobo, and Mubarak Shah. Resolving hand over face occlusion. *Image and Vision Computing*, 25(9):1432–1448, September 2007.

[25] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, 1:I–511–I–518, 2001.

[26] Qing Chen, Nicolas D Georganas, and Emil M Petriu. Real-time vision-based hand gesture recognition using haar-like features. In *Instrumentation and Measurement Technology Conference Proceedings*, pages 1–6. 2007.

[27] ALC Barczak and Farhad Dadgostar. Real-time hand tracking using a set of cooperative classifiers based on Haar-like features. *Res. Lett. Inf. Math. Sci*, 7:29–42, 2005.

[28] R. Lienhart and J. Maydt. An extended set of Haar-like features for rapid object detection. *Proceedings. International Conference on Image Processing*, 1:I–900–I–903, 2002.

[29] Y Freund and RE Schapire. A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting. *Journal of computer and system sciences*, 1997.

[30] Pragati Garg, Naveen Aggarwal, and Sanjeev Sofat. Vision Based Hand Gesture Recognition. *World Academy of Science, Engineering and Technology*, 49(1):972–977, 2009.

[31] Franklin C. Crow. Summed-area tables for texture mapping. *ACM SIGGRAPH Computer Graphics*, 18(3):207–212, July 1984.

[32] B Stenger. Model-based 3D tracking of an articulated hand. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conferenc*, pages II–310. 2001.

[33] N Gupta, P Mittal, and SD Roy. Developing a gesture-based interface. *Journal of the Institution of Electronics and Telecommunication Engineers*, 48(3):237–244, 2002.

[34] MJ Black and AD Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 1998.

[35] Michael Isard and Andrew Blake. CONDENSATION - Conditional Density Propagation for Visual Tracking. *International journal of computer vision*, 29(1):5–28, 1998.

[36] GR Bradski. Computer vision face tracking for use in a perceptual user interface. *Intel Technology Journal*, 1998.

[37] S M Nadgeri, S D Sawarkar, and a D Gawande. Hand Gesture Recognition Using CAMSHIFT Algorithm. *2010 3rd International Conference on Emerging Trends in Engineering and Technology*, pages 37–41, November 2010.

[38] Yuelong Chuang, Ling Chen, Gangqiang Zhao, and Gencai Chen. Hand posture recognition and tracking based on Bag-of-Words for human robot interaction. *2011 IEEE International Conference on Robotics and Automation*, pages 538–543, May 2011.

[39] Yu Ren and Fengming Zhang. Hand Gesture Recognition Based on MEB-SVM. *2009 International Conference on Embedded Software and Systems*, pages 344–349, 2009.

[40] E. Stergiopoulou and N. Papamarkos. Hand gesture recognition using a neural network shape fitting technique. *Engineering Applications of Artificial Intelligence*, 22(8):1141–1158, December 2009.

[41] D. K. Ghosh and S. Ari. A static hand gesture recognition algorithm using k-mean based radial basis function neural network. *2011 8th International Conference on Information, Communications & Signal Processing*, (i):1–5, December 2011.