Sādhanā Vol. 41, No. 2, February 2016, pp. 161–182

Nearest neighbour classification of Indian sign language gestures using kinect camera

ZAFAR AHMED ANSARI and GAURAV HARIT*

Department of Computer Science and Engineering, Indian Institute of Technology Jodhpur, Rajasthan 342011, India
e-mail: gharit@iitj.ac.in

MS received 2 May 2014; revised 3 February 2015; accepted 24 April 2015

Abstract. People with speech disabilities communicate in sign language and therefore have trouble in mingling with the able-bodied. There is a need for an interpretation system which could act as a bridge between them and those who do not know their sign language. A functional unobtrusive Indian sign language recognition system was implemented and tested on real world data. A vocabulary of 140 symbols was collected using 18 subjects, totalling 5041 images. The vocabulary consisted mostly of two-handed signs which were drawn from a wide repertoire of words of technical and daily-use origins. The system was implemented using Microsoft Kinect which enables surrounding light conditions and object colour to have negligible effect on the efficiency of the system. The system proposes a method for a novel, low-cost and easy-to-use application, for Indian Sign Language recognition, using the Microsoft Kinect camera. In the fingerspelling category of our dataset, we achieved above 90% recognition rates for 13 signs and 100% recognition for 3 signs with overall 16 distinct alphabets (A, B, D, E, F, G, H, K, P, R, T, U, W, X, Y, Z) recognised with an average accuracy rate of 90.68%.

Keywords. Indian sign language recognition; multi-class classification; gesture recognition.

1. Introduction

Hand gesture recognition is an old research problem. Ever since the world experienced its first HCI devices, there arose a collective imagination clamouring for contactless control. The movie Minority Report (released in year 2002) eloquently expressed this longing, where 'the user wears gloves with three reflective fingertips to achieve six DOF of hand movements, which is used for some carefully designed gesture metaphors to manipulate data and its layout on the screen' [1].

Gesture recognition means recognition of human gestures by a machine (figure 1). It could be done by using vision algorithms or by physical (glove-based) sensors or both. A human gesture is any meaningful pose or movement of human body that could be used for communication. Gestures are cumulative; they are the sum of all of our body parts' orientations, our expressions and even the contextual setting they are performed in. There are a vast number of human gestures that we use to communicate our ideas. A gesture once performed can perhaps never be repeated exactly by the same person, and the probability that two persons in the world could perform the gesture identically is also extremely low. This becomes a challenge when we attempt to write a recognition algorithm. In the ideal case, a gesture recognition system needs to be portable, lightweight, unobtrusive, robust, fast, low-cost and intelligent enough to understand the emotion and tension of human gestures. The system should not be affected by ambient light intensity and colour of the object to be recognised. The system could give its output in the form of synthesized speech. The system should generalise well to different human body sizes and shapes. Since such a system becomes very complicated, we could break gestures down into simpler subsets like hand gestures (*e.g.* waving), face gestures (*e.g.* smiling) and body gestures (*e.g.* leaning); and try to recognise them separately. Gesture recognition has applications in Natural User Interfaces, Virtual Reality, medical rehabilitation, robot control and even psycho-analytics.

An important application of such a system is human communication itself- *i.e.* sign language. Sign languages are nearly as old as the spoken languages. Plato, in his work Cratylus (fifth century BC), quotes Socrates, "If we hadn't a voice or a tongue, and wanted to express things to one another, wouldn't we try to make signs by moving our hands, head, and the rest of our body, just as dumb people do at present? [2]" Traditional Indian dances like Bharatnatyam strive to tell a tale by the positions of the hands and the body, the fluidity of motion and facial expressions. By some estimates it is more than 2000 years old [3].

Though sign languages have existed in local cultures, standardization began only in the seventeenth century [4].

*For correspondence

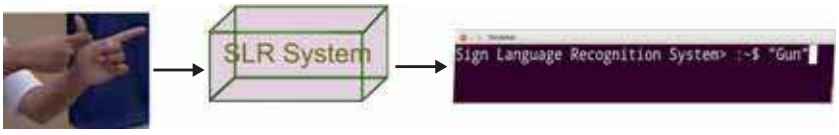


Figure 1. Sign language recognition system.

Sign languages have their own morphological structure and grammatical nuances. They exist in a variety of forms and features. Manual signing or finger-spelling refers to using a sequence of 'alphabet signs' for making an entire word and making a sentence by recursion. In such cases a wide vocabulary is not needed. The more traditional and widely used system has an entry in the sign dictionary for every word. A lot of variations occur in these signs from region to region but a few standardised systems like American Sign Language, British Sign Language, French Sign Language, etc. have emerged. There is also an alternative form of signed communication among the deaf—namely Cued Speech [5]. Cued Speech is not sign language. It is a phonemic speechreading system in which a small number of hand gestures are used in conjunction with speechreading.

Signs may be categorised as being static and dynamic. Static signs involve a fixed hand pose, while dynamic signs involve movement of hands to imply a suggestive motion. Most sign language users use emotion and timing to express themselves better. This makes the language more malleable and hence more comprehensible. Expressions and timing make a world of difference when a non-signer is at the other end of the communication channel.

Every geographical region has developed its own system of signs. Comparatively the Indian Sign Language is much more complex than the American Sign Language. The reasons [6] for this are:

1. Mostly both hands are used in signs.
2. Many of the signs cause occlusions (figure 2).
3. Hand locations with respect to body cause differentiation in signs.

Figure 2. Signs showing occlusions.

Sign language recognition systems have been attempted for American Sign Language [7] and other such languages. There has been lesser such work for Indian Sign Language as ISL has been standardized recently. Moreover, the work has been mostly focussed on a select group of signs namely the manual signs for English alphabets a,b,c and so on. Our work is on a larger vocabulary which has been drawn from three ISL dictionaries – ISL common, ISL technical and ISL banking. Inclusion of such an eclectic mix of daily-use and technical words would lead to a wider utility of the system [8].

Our contribution is the quantitative treatment of the problem of recognition of static gestures of Indian Sign Language. We propose a vote-based feature combination approach for recognition. In the fingerspelling category of our dataset, overall 16 distinct alphabets (A, B, D, E, F, G, H, K, P, R, T, U, W, X, Y, Z) were recognised with an average accuracy rate of 90.68% with our method.

This paper is organized into five parts. Section 2 presents the previous work in this field. Section 3 gives details about the dataset being used in this work, the denoising procedures and the segmentation processes used. Section 4 gives the details of the features extracted and the recognition rates achieved. Section 5 sums up the discussion and lists possible future work.

2. Literature review

Mitra & Acharya [9] provide an excellent review of the work on gesture recognition, including facial gestures. Bilal *et al* [10] present a study on vision-based static hand shape detection and recogntion techniques. We organize the literature survey in the following subsections according to techniques which take input from RGB camera, Gloves, and depth camera, respectively.

Tables 1, 2, and 3 list the existing techniques proposed for Indian Sign Language, other sign languages, and some commonly used gestures respectively.

2.1 *RGB based approaches*

Quek & Zhao [11] have used disjunctive normal forms to build a simple learning system for hand poses. The system reduces the number of features to be computed at recognition time by choosing a subset of the original feature set.

Table 1. Approaches for Indian sign language.

Authors	Dataset/sensor	Segmentation/ features/ recognition method	Recognition rate (%)
Geeta [6]	static (29, 1450) RGB	B-Spline, key maximum curvature points; SVM	≈90
Rekha [14]	static (23, 920) dynamic(3, 66) RGB	Skin colour, edge orientations, texture; SVM	77.2
Joyeeta [15]	static (24, 240) RGB	Skin colour, Eigen values; nearest neighbour	97
Bhuyan [16]	static (8, 400) RGB	Skin colour, geometric features; nearest neighbour	>90

SVM, Support vector machines.
The pair of integers in parentheses indicate the number of classes(signs) considered and the size of the training dataset used.

Table 2. Approaches for other sign languages.

Authors	Dataset/sensor and gesture type, sign language	Segmentation/ features/ recognition method	Recognition rate (%)
Starner [12]	dynamic (40, 500) RGB, ASL	Skin colour, second moments of area, angle of axis etc.; HMM	92
Luis-Pérez [13]	static (23, NA) RGB, MSL	Active contours, shape signatures; neural networks	95.8
Hernandez [18]	static (26, 1300) Accele Glove, ASL	Hand shape, palm orientation positions; decision trees	> 90
Liang [19]	static (51,613), dynamic (250, 648) DataGlove, TSL	Dynamic (sentence level recognition); Azimuth, elevation, roll of palm, joint positions	static(95%), dynamic (80.4%)
Pugeault [7]	static (24, 48000) Kinect, ASL	Thresholding, Gabor filtering; Random forests	75
Keskin [24]	static (10, 30000) Kinect, ASL	Hand model, Shotton features [26]; Random forests	99.9
Saengsri [20]	dynamic (16, 64) 5DT Data Glove 14 Ultra, ThSL	Neural networks	> 90

MSL, Mexican sign language; TSL, Taiwanese sign language; ASL, American sign language; ThSL, Thai sign language; *HMM*, Hidden Markov models.

Table 3. Approaches for common gestures.

Authors	Dataset/sensor and gesture type	Segmentation/ features/ recognition method	Recognition rate (%)
Quek [11]	dynamic (20, 1842) RGB, Common gestures (up/down etc.)	Area of the bounding box, centroid of hand, principal axis etc.; DNF	94.36
Kenn [17]	static (12, NA) Mechanised Glove, Simple gestures	Pitch and Roll from gravity vectors	NA
Bergh [21]	dynamic (6, 350) RGB and ToF cameras, Common gestures	Skin Colour; ANMM	99.54
Ren [25]	static (10, 1000) RGB, Common gestures	FEMD, skin colour; Nearest neighbour	93.9

FEMD, Finger-Earth Mover's distance [25]; ANMM, Average neighbourhood margin maximization.

They use 15 classes and skin colour-based segmentation to get 94.36% recognition rate.

Starner *et al* [12] recognised, in what is perhaps the earliest work on sign language, an ASL (American Sign Language) vocabulary of 40 words using a single colour camera on unadorned hands based on skin colour. They use HMMs to recognise signs in two settings: a headcap-mounted camera and a desk-mounted camera. They locate the hand by dilation around a seed point found using skin colour intensity, and calculate features by taking the second moment of the segmented blob.

Luis-Pérez *et al* [13] recognised 23 alphabets of the Mexican Sign Language to control a service robot. They use snakes to segment the RGB image, shape signatures as features, and neural networks for recognition. They achieve 95.8% recognition rate. The recognition is then used to direct a robot to do certain tasks.

Geetha & Manjusha [6] use B-Spline approximation to recognise a subset (29 signs) of static ISL (Indian Sign Language) signs. They trace the boundary of the hands using connected components and then find the points with curvature greater than a threshold. Then a B-spline curve is fit to these points and features are extracted to train a multiclass SVM.

Rekha *et al* [14] work on a vocabulary of 23 static signs and 3 dynamic signs of ISL. They use skin colour segmentation through a Gaussian model to locate hands. Then they use edge orientations (from Principle Curvature based Region Detector) and texture (from Wavelet Packet based Decomposition) as features to train a multiclass SVM and get a recognition rate of 86.3%. However their approach is very slow and has an average recognition time around 55 s.

Singha *et al* [15] use Eigen values extracted from segmented hands to classify 24 static signs of ISL using RGB images. They do a nearest neighbour classification using eigen value weighted Euclidean distances and achieve 97% recognition accuracy.

Bhuyan *et al* [16] use a skin colour based segmentation procedure to extract hands. A model based approach with geometric features (relative angles between fingers) and Homogeneous Texture descriptors (HTD) is then used to classify static signs according to a nearest neighbour heuristic. They use eight gestures with a training dataset of 400 images and get a high recognition accuracy (above 90%).

2.2 *Glove based approaches*

Kenn *et al* [17] developed a glove based user interface device using a context interface. They use a glove with integrated electronics to sense acceleration and pose of the hand. This glove sends its sensor data wirelessly to a computer where it is filtered. Six gestures were supported initially with the option to learn and override as the contextual information in a context repository grows. The recognised gestures are used to send instructions to an application like map navigation or robot control.

Hernandez-Rebollar *et al* [18] recognise 26 hand shapes of the ASL by the help of an Accele Glove. They use hand shape and palm orientation as features and obtain classification by a discrimination-based heuristic. They achieve very high recognition rates but their classification criteria is hand-tailored to the symbols that they considered and hence it is not generic.

Liang & Ouhyoung [19] use DataGlove to build a sign language interpreter (on dynamic as well as static signs) on a vocabulary of 250 words from Taiwanese Sign Language. Azimuth, elevation, roll of palm and glove data about joints are used for recognition. They use HMMs to achieve sentence level recognition rate of 70.4% and 89.5% for short and long sentences respectively.

Saengsri *et al* [20] use the 5DT Data Glove 14 Ultra with 14 sensors and a motion tracker to provide orientation and position of the hand. They use Neural Networks for classification. Their dataset is composed of 64 instances from a vocabulary of 16 signs. They achieve a high recognition accuracy above 90%.

2.3 *3D based approaches*

Van den Bergh & Van Gool [21] use a hybrid approach to track hands by combining RGB and depth data obtained from a ToF camera. They use a face detector (implemented in OpenCV) to locate the face in the RGB image and obtain its distance in the depth image, and for the remaining regions which satisfy a depth threshold, they identify the skin colour using a Gaussian Mixture Model based skin colour model. This enables them to find anything that is extended (like hands) in front of the body towards the camera. After detecting the hands, they use Average Neighbourhood Margin Maximization (ANMM) [22] (approximated using Haarlets) to do matching from their database. They achieved a recognition rate of 99.54% on a vocabulary of six symbols and 350 sample images.

Argyros & Lourakis [23] use stereo cameras to detect hands and associate them in temporal space to their 3D reconstruction. They use 2D trackers to segment blobs of hands on the basis of skin colour segmentation. Then they match their shapes through contour alignment by a variant of Iterative Closest Point algorithm. Consequently, triangulation results in the 3D reconstruction of the matched hand contours.

Pugeault & Bowden [7] build a real-time recognition system for the alphabets of ASL using a range sensor Kinect. They use Gabor filters and multi-class random forests to achieve high classification rates.

Keskin *et al* [24] fit a skeleton in real-time to the hand to recognise digits of ASL by using an object recognition by parts approach. They too use the range sensor Kinect.

Ren *et al* [25] use thresholding for segmentation of hands from Kinect data. They detect fingers and use Finger-Earth Movers Distance (FEMD) to differentiate between gestures. They use a vocabulary of 10 gestures and a total of 1000 images.

Figure 3. Sample IR image (colour coded) for the word 'Bottle'.

In a related work, Shotton *et al* [26] tackle the problem of human body posture recognition through a machine learning approach. They use relatively weak features, but leverage their full power by combining them and using randomized decision forests. They were successful in generating 3D joint proposals in real-time with Kinect. Their work has been used in the gaming technology of Microsoft XBox which employs these 3D joint proposals to detect players' gestures. However, the hand skeleton is not a part of the generated skeleton and sign languages require hand pose estimation.

Indian Sign Language recognition has not received the type of attention ASL or BSL has received. Most of the experiments have been done on very small datasets and assumed certain restraints on the type of surroundings or the type of signs (single-handed or open/closed palm-only signs). Usage of special gloves with built-in sensors to detect hand postures makes the recognition system less user-friendly and more expensive. Kinect acquired input has colour and depth information. Recognition system using depth information has not been developed for Indian Sign Language.

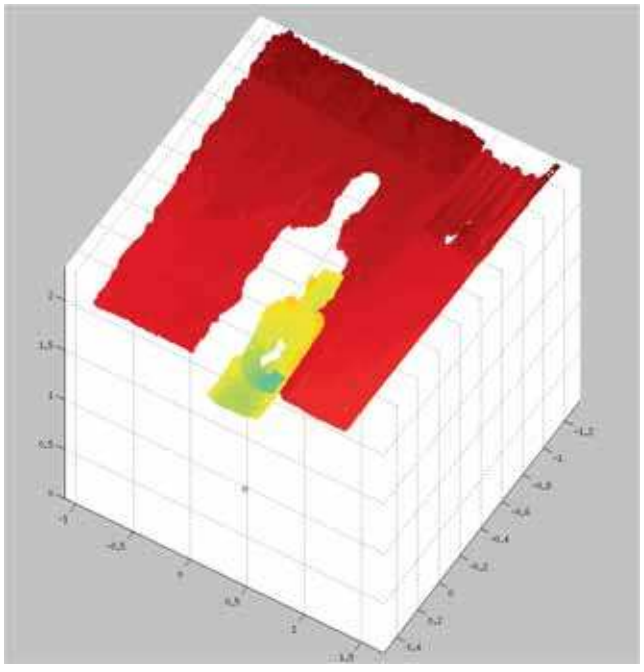


Figure 4. 3D reconstruction of a scene by depth image. The black circle represents the position of the Kinect.

A single depth image acquired using Kinect consists of pixels with integer values in the range [0, 2047]. This depth value is somewhat proportional to the distance of the corresponding object point from the depth sensor (figure 3). Points that are closer than 0.8 m and farther than 3.5 m are reported with a depth value of 2047 [27].

The RGB camera on Kinect provides a frame rate of upto 30 Hz and a resolution of upto 640 × 480. The camera gives two video streams simultaneously, namely an RGB video stream from its VGA sensor and a depth stream from its infrared sensor (figure 4).

3. Preprocessing and segmentation

Acquisition of an image dataset requires a well thought-out process. Clues from previous such instances are helpful. However, there have been no stereo image datasets for Indian Sign Language to the best of our knowledge. The language consists of thousands of documented and undocumented signs. Therefore, a carefully selected subset of gestures from the ISL dictionary was used for the task (table 4).

3.1 *Dataset used*

The dataset (see table 5 for sign names), henceforth referred to as 'the ISL Dataset', consists of 140 static signs hand-picked from ISL. Volunteers were advised to be upright and keep their hands distinct from the body as much as possible. This was done to enable depth images to have a greater level of detail. Right and left hands were not interchangeable in all signs. All volunteers were non-native to sign language. Volunteers were advised to reform the signs after each trial of a sign. This ensured non-duplicacy of the image. Volunteers stood about half a metre from a flat surface. There was a distance of approximately 1 m between the Kinect and the volunteer. The dataset has been selected from the ISL general, technical and banking dictionaries. Apart from complete words, the dataset also has signs for manual fingerspelling (signs for alphabets). A word may have different variants—particularly single-handed and two-handed variants. In such cases all have been included. Symbols for single-handed 'J', single-handed 'Z' and double-handed 'J' are dynamic in nature, so they have not been included. Some

Table 4. The ISL dataset.

Category	RGB images	Depth images
In 1 class	36	36
In all classes	5041	5041

Number of classes: 140.
Number of volunteers: 18.

Table 5. List of signs of the ISL dataset.

1. One	2. Two	3. Three
4. Four	5. Five	6. Six
7. Seven	8. Eight	9. Nine
10. Ten	11. A	12. Add
13. Appreciation	14. A-SingleHanded	15. Assistance
16. B	17. Bell	18. Between
19. Bhangada	20. Bite	21. Blow
22. Bottle	23. bowl	24. Boxing
25. B-SingleHanded	26. Bud	27. C
28. Conservation	29. Control	30. C-SingleHanded
31. D	32. Density	33. Deposit
34. D-SingleHanded	35. E	36. Elbow
37. E-SingleHanded	38. F	39. Few
40. Fine	41. Friend	42. F-SingleHanded
43. G	44. Ghost	45. Good
46. Gram	47. G-SingleHanded	48. Gun
49. H	50. Handcuffs	51. Help
52. Here	53. Hold	54. How
55. H-SingleHanded	56. I	57. Intermediate
58. Iron	59. I-SingleHanded	60. It
61. K	62. Keep	63. K-SingleHanded
64. L	65. Leaf	66. Learn
67. Leprosy	68. Little	69. Lose
70. L-SingleHanded	71. M	72. Mail
73. Me	74. Measure	75. Mirror
76. M-SingleHanded	77. N	78. Negative
79. N-SingleHanded	80. O	81. Obedience
82. Okay	83. Opposite	84. Opposition
85. O-SingleHanded	86. P	87. Participation
88. Paw	89. Perfect	90. Potentiality
91. Pray	92. Promise	93. P-SingleHanded
94. Q	95. Q-SingleHanded	96. Quantity
97. Questions	98. R	99. Respect
100. Rigid	101. R-SingleHanded	102. S
103. Sample	104. Season	105. Secondary
106. Size	107. Skin	108. Small
109. Snake	110. Some	111. Specific
112. S-SingleHanded	113. Stand	114. Strong
115. Study	116. Sugar	117. T
118. There	119. Thick	120. Thursday
121. T-SingleHanded	122. U	123. Unit
124. Up	125. U-SingleHanded	126. V
127. Vacation	128. Varanasi	129. V-SingleHanded
130. W	131. Warn	132. Weight
133. Work	134. W-SingleHanded	135. X
136. X-SingleHanded	137. Y	138. You
139. Y-SingleHanded	140. Z	

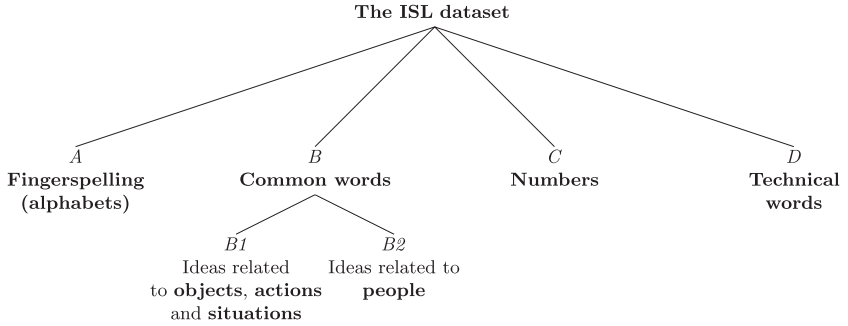


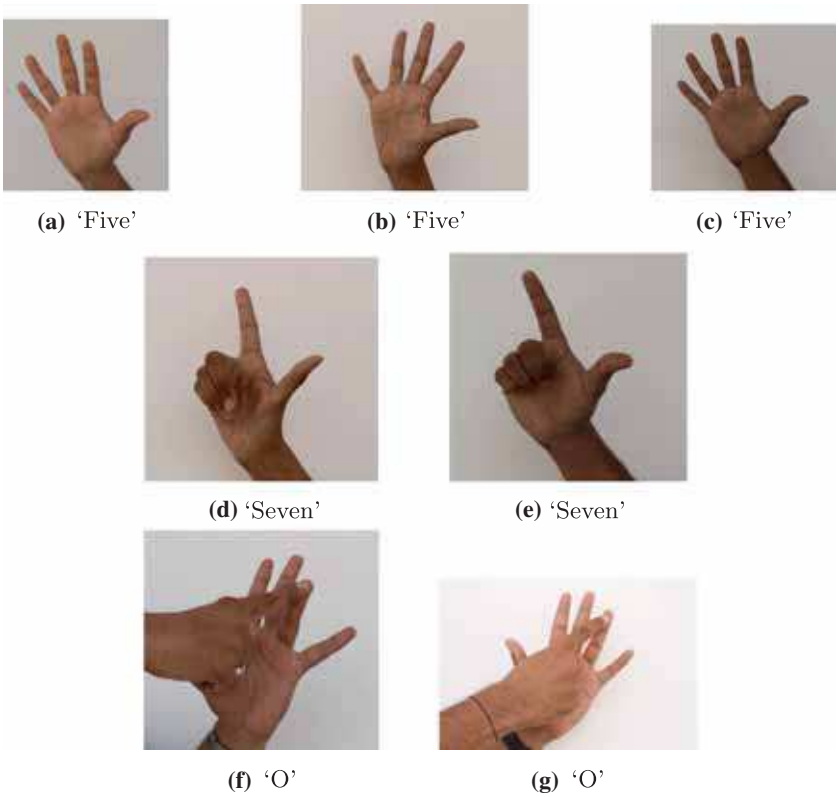
Figure 5. Dataset categorisation. *Italicised* letters denote category names.

Hindi words like 'Bhangada' and 'Varanasi' have also been incorporated. Some images were rejected as the gestures in them were incorrectly performed.

3.1a *Dataset categorisation*: This 140 class ISL Dataset was divided into subsets on the basis of their utility. The categorisation is shown in figure 5 and table 6.

Table 6. Dataset categorisation.

Designation	Category	Classes in this category
A	Fingerspelling (alphabets)	14, 11, 16, 25, 27, 30, 31, 34, 35, 37, 38, 42, 43, 47, 55, 49, 56, 59, 61, 63, 64, 70, 71, 76, 77, 79, 80, 85, 86, 93, 94, 95, 98, 101, 102, 112, 117, 121, 122, 125, 126, 129, 130, 134, 135, 136, 137, 139, 140
B1	Ideas related to objects, actions and situations	17, 18, 20, 21, 22, 23, 26, 28, 29, 32, 45, 46, 48, 50, 52, 54, 58, 57, 60, 65, 68, 72, 74, 75, 78
B2	Ideas related to people	41, 19, 24, 33, 36, 44, 51, 53, 62, 66, 67, 69, 13, 73, 81, 87, 91, 92, 97, 107, 113, 115, 127, 131, 133, 138, 12, 15, 82, 105, 110, 108, 111
C	Numbers	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
D	Technical words	28, 29, 32, 57, 78, 84, 96, 100, 106, 111, 123, 132, 74, 33, 90, 58



(a) 'Five' **(b)** 'Five' **(c)** 'Five'

(d) 'Seven' **(e)** 'Seven'

(f) 'O' **(g)** 'O'

Figure 6. Different types of intra-class variations. (**a**), (**b**), (**c**) show shape variations. (**d**), (**e**) show rotation variations. (**f**), (**g**) show orientation variations.

3.1b *Challenges of this dataset vis-à-vis the problem statement*: Challenges of this dataset are:

1. Requirement of robust scale, orientation and rotation invariant features:

   In this dataset, for images in the same class, the volunteers have changed orientations and rotated their signs. The signs have been taken at differing distances from the camera. All these factors have made it necessary that the features used in recognition be robust to scale, orientation and rotation (see figure 6).
2. Requirement of fast segmentation (for real-time performance):

   The recognition system needs to be fast enough to serve a real-time gesture analysis.
3. Fast removal of high image noise:

   The depth stream of Kinect is infested with a lot of flickering and random noise. Most of this noise is due to the infrared laser's scattering on incidence on an abject. Also the Kinect gives a depth value of 2047 for an object beyond its sensing limit. For detection of good keypoints, it is highly important to remove the noise.
4. Tackling high intra-class variation:

   The dataset must ensure that a lot of intra-class variation is captured.
5. Tackling low inter-class variation (similar gestures but different classes):

   For a classification task it is better if the classes are highly distinct. However, there are many sign classes in this dataset which are very similar in terms of hand positions and shapes (see figure 7). In actual usage, such signs are resolved using the context of the sentence. However, we are not attempting to correlate the context.
6. Generalisation for a high range of human body types in terms of shapes and sizes:

   There is a vast range of human body types, shapes and sizes. The system should be able to scale well to variations in hand shapes and sizes. There are 27 degrees of freedom [28] for a human hand and hence its flexibility makes a great number of variations possible for the same sign.
7. Two-handed signs: Two-handed signs are more complex to track than single-handed signs.

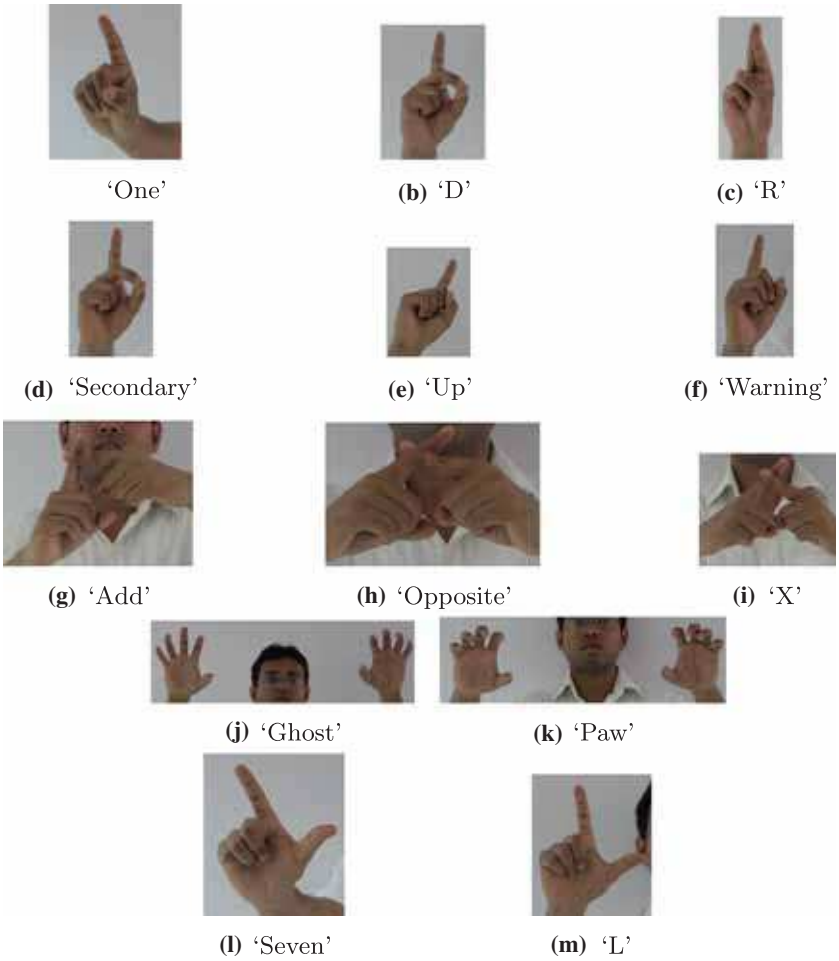


Figure 7. Similar signs. (**a**)–(**f**); (**g**)–(**i**); (**j**), (**k**); and (**l**), (**m**) are similar yet belong to different classes.

3.2 *Assumptions used*

1. There is no object between the user and the Kinect.
2. Kinect has been placed roughly at the level of mid-torso of the user.
3. The user is standing atleast 1.5 m away from the Kinect.
4. The user is standing closer than 3 m from the Kinect.
5. The user is standing with a minimum distance of 0.5 m from his/her nearest object from behind.
6. Kinect is not tilted upwards or downwards, but is facing the horizon.
7. User's hands should be as distinct as possible from the rest of the torso.
8. The system has been setup indoors away from direct sunlight.
9. Bangles, wristwatches and other such appendages should be removed before the system's usage.
10. Preferably, user's background should be clutterless.
11. There are not more than one humans in the view of Kinect.

3.3 *Systems and platforms used*

The recognition system was implemented on an Intel Core *i*5 CPU (2.80 GHz $\times$ 4) with 8 GB RAM. The system ran Ubuntu 12.04 (32 bit). The system was implemented in C++ programming language. Matlab R2011b was used for analysis and dataset archiving. 3D modelling tools like MeshLab were used for visualising the data. Libraries of OpenNI [29] and OpenKinect (http://openkinect.org/wiki/Main_Page) (Libfreenect), PCL (Point Cloud Library) [30] were used for acquiring and handling depth data from Kinect.

3.4 *Denoising the image*

The depth image is noisy both in spatial and time domains. The noise is represented in the form of jagged edges and 2047 pixel values (black spots). For denoising we need a fast, efficient filter that could remove most of the 2047 values without affecting silhouettes. Most of our features are extracted from the hand silhouettes. If the edges in the depth image are degraded or flattened too much, the features extracted might be ambiguous and insufficient for classification.

It is important to get rid of most 2047 pixel values as they interfere in hand segmentation. This preprocessing should also be fast enough. Kinect gives video at a rate of about 30 FPS. Techniques like non-local denoising [31] are not yet fast enough for real-time performance.

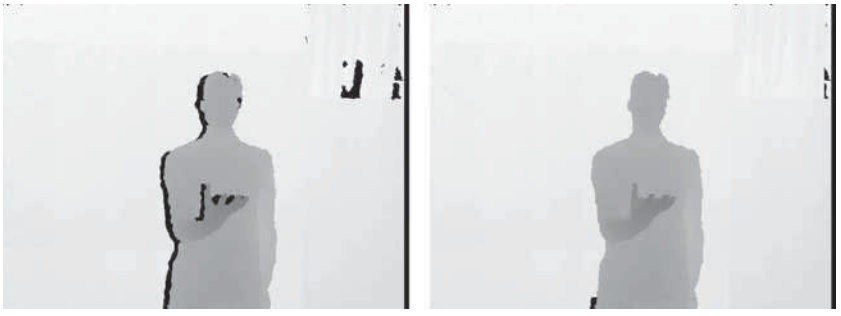
Mean filters blur the data and much of the detail is lost. In this case details like finger shapes would be of critical importance in later recognition. Median filtering [32] is typically useful in such cases. We have used median filtering to get rid of the spiky noise and to preserve the edge information (figure 8).

3.5 *Segmentation*

Having denoised the dataset images, it is now imperative that we proceed to segmentation of those parts of the images which are useful for recognition of the gestures. Since we are dealing with static signs, hands of a person are easily the most discerning parts of the image that should be concentrated on. Hands up to elbows, in most cases, exhibit ample evidence for classification given the classes themselves are distinct enough. Segmentation must be done fast enough for a real-time performance.

Since the system needs to be invariant of lighting conditions, only the depth image is used for localisation of hands. We have assumed (see Section 3.2) that there are no objects between the user and the Kinect. Mostly the *hands lie closer to the camera* than the rest of the body. Once the depth image has been segmented into cogent segments, we can choose the segment that is *closest* to the camera to identify the hands.

3.5a *Normalised cuts*: Shi & Malik [33] gave a segmentation algorithm based on a graph-theoretic treatment to the perceptual segmentation problem. They use a global criterion (minimisation of the normalised cut) to effectively segment an image. However, this approach to graph partitioning is NP-complete. Hence, an approximate solution is found using optimisation packages. We used this algorithm to segment out depth images. This method was not deemed



(a) Noisy image **(b)** Denoised image

Figure 8. Image before and after denoising. Darker colour represents lower depth of the corresponding pixel.

Table 7. Time taken for N-Cut segmentation.

Level of downsampling %	Average time taken for the dataset (s)
80	1.6357
65	6.0847

useful for our purpose as the images had to be downsampled drastically in order to get even close to the time complexity that our system requires (see table 7).

3.5b *K-means clustering*: *K*-means clustering is an unsupervised learning algorithm. We used the *city-block distance* as the distance metric (as it is faster) and, assuming all the assumptions were followed (see Section 3.2), we got faster and better hand segmentation (see figure 9).

Speeding up of K-means clustering: Random initialisation of the *K*-means algorithm involves initializing the *K* cluster centres with a random 3D depth. Pseudo-random initialisation leads to unpredictable clustering time intervals. Therefore, it was necessary to initialise *K*-means some other way so that it becomes more balanced in performance. We designed a novel algorithm to achieve this. There are a number of local maxima in the histogram of depth values. These local maxima show the approximate depths of major objects in the space in front of Kinect. So we calculated the depth values which were local maxima by our algorithm and then initialised the *K* cluster centres with *K* of those local maxima which had the highest frequencies.

Our algorithm *automatically* chooses (see figure 17) the initial *K* seed values as the most 'prominent' and 'distinct' local maxima in the current histogram of depth values, so as to speed up the process of clustering. We do not want those local maxima which are too close or too small. We want 'distinct' and 'high frequency' points which could represent distinct object densities like hands, torso and background. The algorithm is summarized below in three steps:

1. Find simple local maxima using the histogram, governed by PEAK_MIN_DEPTH and SEARCH_RANGE.
2. Grow regions (governed by tuning parameters PEAK_MIN_BRACKET) around these local maxima.
3. Select the largest depth in these regions as the final initialisation seed points.

Increasing PEAK_MIN_BRACKET will make the regions larger and the resultant seeds may be very close. PEAK_MIN_DEPTH acts as a threshold on the minimum frequency that a depth point must have in order to be eligible for being selected as a local maximum. Increasing SEARCH_RANGE increases the area where we compare adjacent simple local maxima to find the more dominant ones in step 1, decreasing the total number of maxima found at the end. We have empirically set the values for the parameters PEAK_MIN_DEPTH to 100, SEARCH_RANGE to 70, PEAK_MIN_BRACKET to 25. After initialising with the local maxima, the *K*-means tends to become faster and more consistent in average runtime. We then select the closest cluster on the basis of the clusters' mean points' depth.

4. Feature extraction and recognition

Having extracted the segments containing the hands from the depth images, the next step is to extract appropriate features. It is desirable that the feature matching between the training set images and the test set image be done in real time. Possible choices of features include silhouettes, depth profiles, relative positions of hands to each other, etc. In order to compute features, it is first often necessary to detect those points which are good locations for extracting the feature descriptors. Feature detection is a Boolean operation carried out at each point in the image to find whether that point is a good candidate for describing a distinguishing property of that image for future recognition.

Detected points are then used to define feature vectors. The feature detection stage might also give useful cues like edge orientation. Features thus extracted from the training images are indexed in a *k*-d tree, where *k* is the dimensionality of the feature vector. When the system runs, the test image is received from the user, preprocessed, segmented and features are extracted from it. Then a nearest neighbour

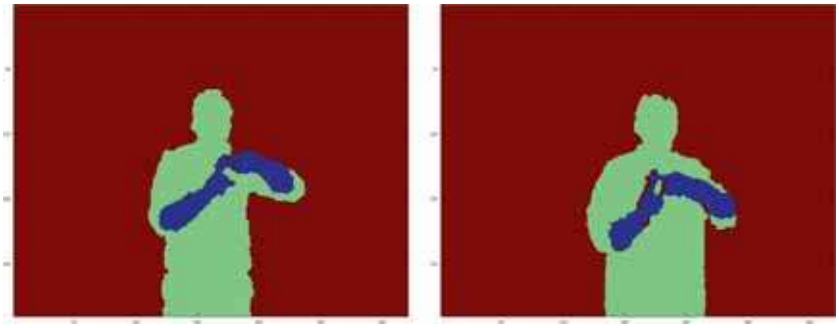


Figure 9. Sample colour-coded clusters ($K = 3$). Here blue coloured regions show segmented hands.

search is run to find the point(s) in the index that, according to some recognition heuristic (described in Sections 4.1b, c; 4.2b), is/are chosen as the output.

4.1 *Features and recognition*

In this section, we describe the division of dataset, features extracted and the recognition schemes used in our approach.

4.1a *Training and test datasets*: We divide the test and training datasets such that on an average 5% of total samples (for all classes) are in the test dataset and the rest are in the training dataset. To thoroughly check our results, we have chosen all test images to be of a *new person* each time, whose images are not present in the training dataset. This shows how well our system will adapt to a new person's gestures.

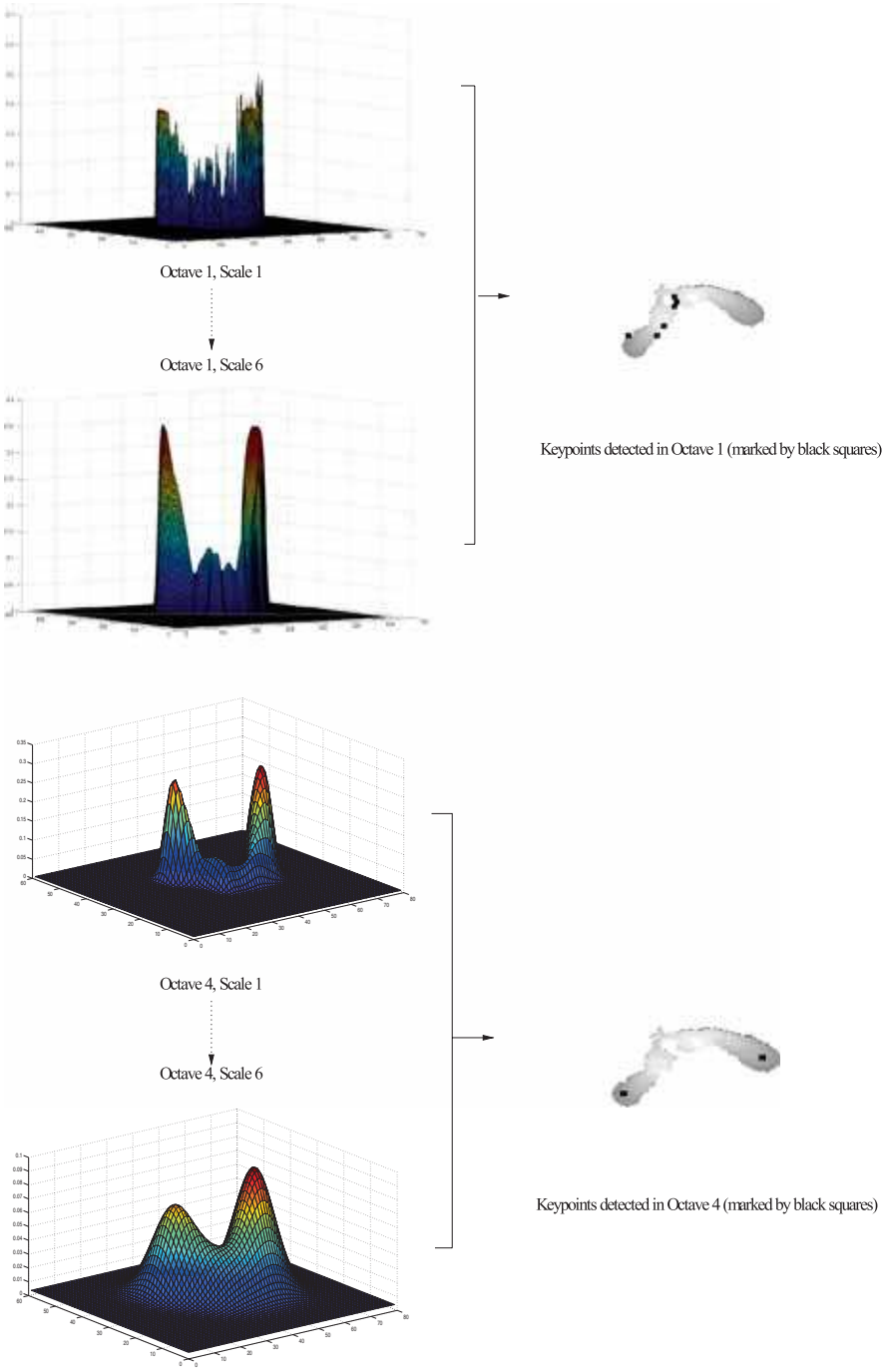


Figure 10. The depth image is smoothed with Gaussians of increasing sigma, and it can be observed in the depth plots (on the LHS) that low level features are extinguished by the time we reach Octave 4. Hence, high level keypoints are detected in Octave 4 while low level keypoints are detected in Octave 1 (on the RHS).

We repeat the experiment for 17 times, each time picking a different person and report the average recognition rate.

4.1b *SIFT*: Scale invariant feature transform (SIFT) was developed by Lowe [34]. The SIFT descriptor is invariant to translations, rotations and scaling transformations and efficiently handles moderate perspective transformations and illumination variations. Since our dataset has a lot of such variations (see figure 6 for examples of rotation and orientation variations) SIFT seemed a good candidate feature.

SIFT was originally developed for grey level images. The original paper described a method to detect interest points using local gradient directions of image intensities. These interest points were then used for computing statistical features in the local neighbourhood around the interest point. The SIFT feature has been extended to colour images and 2+1 D spatio-temporal data [35].

Keypoints: A Gaussian mask with a standard deviation of sigma s, when convolved with an image I, blurs its features. The greater the value of s, the greater is the smoothing effect and the more 'high level' features remain after the convolution. This is because the Gaussian kernel, as s increases, has increasing weightage of neighbouring pixels' contributions in the convolution. Interest points (not necessarily edge points) which define unique features of an image are obtained by finding local extrema of the scale-normalized Laplacian of Gaussian (LoG) of that image [36, 37]. The reason we presmooth with the Gaussian is that the Laplacian is very sensitive to noise. Smoothing with the Gaussian is helpful in removing features that are too low-level, or in other words, this pre-processing step reduces the high frequency noise components before taking the Laplacian (see figure 10). Scale-normalised Laplacian is defined as

$$\begin{aligned}\nabla^2_{norm} L(x, y; s) &= s(L_{xx} + L_{yy}) = s\left(\frac{\partial^2 L}{\partial x^2} + \frac{\partial^2 L}{\partial y^2}\right) \\ &= s\nabla^2(G(x, y; s) * f(x, y)) \qquad (1)\end{aligned}$$

from smoothed image values $L(x, y; s)$ computed from the input image $f(x, y)$ by convolution with Gaussian kernels G.

$$G(x, y : s) = \left(\frac{1}{2\pi s}\right) e^{-\frac{x^2+y^2}{2s}},$$

where s is the scale or the variance of the Gaussian kernels.

Then the scale-space extrema are detected from the points $(x, y; s)$ in scale-space at which the scale-normalized Laplacian assumes local extrema with respect to space and scale.

This can be done in a discrete setting by using the DoG (Difference of Gaussians) operator which is an approximation of the Laplacian operator (see figure 11).

$$\begin{aligned}DoG(x, y; s) &= L(x, y; s + \Delta s) - L(x, y; s) \\ &\approx \left(\frac{\Delta s}{2}\right)(\nabla^2 L(x, y; s)). \qquad (2)\end{aligned}$$

If $\sigma_{i+1} = k * \sigma_i$ then

$$DoG(x, y; s) \approx \left(\frac{k^2 - 1)}{2}\right) \nabla^2_{norm} L(x, y; s). \qquad (3)$$

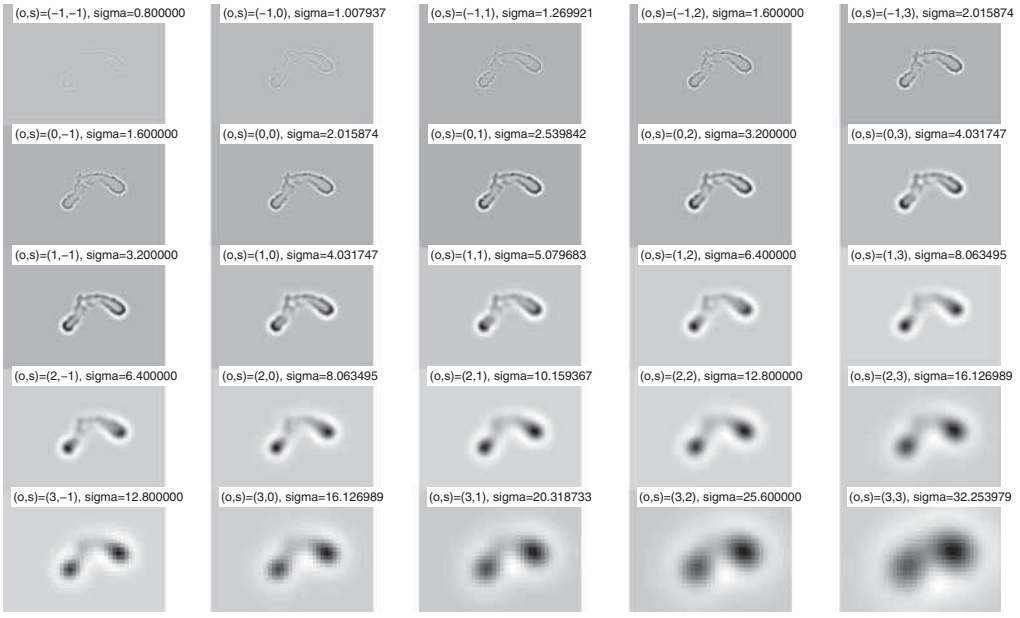


Figure 11. The figures show the DoGs obtained from the Gaussians of all octaves, parts of which are shown in figure 10. Each octave subsamples the image used in the previous octave.

We use a range of such sigma values say, s, ks, k^2s,..., k^ns for the Gaussian kernels and blur the original image with them. Now these images, in increasing values of sigma, are stacked onto one another to get a 3D stack of pixels (x, y axes for a single image's pixels, and z-axis for sigma values). These images represent Gaussian smoothed images, and we subtract one image $G(x, y, s)$ from the image above it $G(x, y, ks)$ to get another set of images representing the DoG. In order to find the extrema of this DoG, in this 3D stack (octave), we sample a $3 * 3 * (n + 1)$ cuboid, and find such pixels which have their immediate neighbours (26 pixels which are above, below and on the sides) either larger or smaller than themselves. These points then become interest points. The interest points are preserved under scaling transformations. The selected scale levels are transformed in accordance with the amount of scaling. The DoG is a computationally efficient approximation [38, 39] to the LoG operator.

It is worth mentioning the reasons behind the choice of a Gaussian filter for smoothing. It is to be ensured that the smoothing does not give rise to new structures at coarse scales. In the scale-space literature [40, 41], a set of axioms (scale-space axioms) [42] have been proposed to formulate requirements for a filter for such situations, and it has been concluded that the Gaussian scale space is the canonical way to generate a linear scale space [43, 44]. The scale space axioms have shown that the Gaussian kernel is unique in the sense that it is scale invariant, rotation invariant, shift invariant and does not enhance local extrema.

Descriptor: The SIFT descriptor is a statistical local neighbourhood feature descriptor. The size of the neighbourhood is normalized in a scale-invariant style. To make it rotationally invariant, we calculate the dominant orientation in the neighbourhood. This dominant orientation is used to align the grid over which histograms are calculated. The gradient orientations are quantized into eight units [35]. This leads to a 128 dimensional feature vector for each interest point (figure 12).

Recognition: Lowe recommends counting those points as matches for which the ratio of distances to the nearest and the next nearest points is less than 0.8 (figure 13). Recognition was achieved using indexing in k-d trees.

We divided our dataset into training and test sets iteratively (17 times), each time picking a different set of images from our master dataset. Then for each image in the training set, we calculated the SIFT feature vectors, and indexed them in a k-d tree along with their class labels. After the training, we calculated feature vectors for each test image, and found the five nearest neighbour feature vectors (according to Euclidean distance), and their corresponding classes, by searching the k-d tree.

We calculated the votes for each feature vector in favour of a particular class. For calculating the vote of a feature vector, we tried 4 different strategies:

1. Simple majority:
   We found the feature class with the maximum occurrence among the five nearest classes. In case of a tie, the one closer to the test feature was counted.
2. Simple weighted majority:
   We found the feature class with the maximum weighted vote among the five nearest classes. For each class the weight was inversely proportional to its corresponding feature's Euclidean distance from the test feature, as follows.
   $$w_i = \frac{1}{d_i}$$
   where w_i is the weight of the i^{th} feature's vote to its class and d_i is its distance from the test feature.
3. Exponentially weighted majority:
   We found the feature class with the maximum exponential weight among the five nearest classes. For each class, the weight was calculated as
   $$w_i = e^{-a*d_i}$$

Figure 12. SIFT keypoints in 3D. Green dots show the SIFT keypoints extracted according to [45] in the 3D reconstructed model.

Figure 13. SIFT feature matching. Green lines indicate the interest points that satisfy the Lowe criteria.

Table 8. Average SIFT recognition rates for 17 iterations. All recognition heuristics are shown. Exponentially weighted majority is used with $a = 100$.

Category	Average recognition rate (%)
Simple majority	44.03
Exponentially weighted majority	49.36
Simple weighted majority	48.91
Nearest neighbour	49.07

where d_i is the Euclidean distance of the ith feature from the test feature and a is a constant.

4. Nearest neighbour:
   We found the feature class which was closest to the test image feature.

After calculating the class predicted by each feature of a test image, we apply the final recognition heuristic. This we do by finding the class with the maximum occurrence among the predicted classes by all features for that particular image. In case of a tie, all such classes are reported as possible matches.

This is done for all iterations and average recognition results thus obtained are shown in table 8. Exponentially weighted majority heuristic performs best with $a = 100$.

4.1c *Recognition using Viewpoint Feature Histogram (VFH)*: As we have seen in the case of SIFT, for each image we get a number of feature vectors which complicate the process of matching afterwards. Viewpoint Feature Histogram (VFH) [46] descriptor is a robust viewpoint-invariant descriptor used for extracting features from 3D point clouds. VFH can be used for both object and pose recognition. The computational cost for VFH recognition is low and it is ideally suited for recognition in real-time applications. VFH seems to work on noisy and cluttered data effectively.

Descriptor: Although the descriptor was designed for rigid objects, we use it here to recognise our signs which are prone to deformations. The descriptor consists of a *single* 308 dimensional vector for a depthmap. This makes the job of matching fairly straightforward.

For a given radius r, the feature will first estimate the surface normals at each point p by performing Principal Component Analysis (PCA) on the k-neighbourhood defined by a sphere centered at p with radius r. It estimates features (which are a measure of the angles between the points' normals and the distance vector between them) between every point and its k nearest neighbours, followed by a reweighting of the resultant histogram of a point with the neighbouring histograms. This descriptor makes use of FPFH (Fast Point Feature Histograms) [47], extending it to a single feature vector for a given cluster of points in 3D by incorporating the viewpoint direction into the relative normal angle calculation. The computational complexity of VFH is $O(n)$ [46] (figure 14).

Recognition: Fast approximate k-NN matching is done by using a k-d tree. We divided our dataset into test and training sets multiple times, each time picking different images. We first converted our images into 3D clusters. We then calculated VFH features for each of our training clusters, and indexed them in the form of a k-d tree, along with their class labels. We then calculated the VFH features on 3D clusters of the test images. We calculated the 15 approximate nearest neighbours according to Chi-square distance criterion of the VFH feature of our test image. We could not take so many in the previous case of SIFT because there were multiple feature vectors for a single image. We followed the following approaches for recognition:

1. Top nearest neighbour:
   The nearest neighbour to the test feature was chosen as the recognition result.
2. Simple majority:
   The maximally occurring class among the 15 nearest neighbours was chosen as the recognition result. In case of a tie, one with the lesser distance from the test feature was chosen.
3. 50% majority:
   The class which commanded half of the total votes of the 15 neighbours was chosen. In the absence of such a class, no recognition result was shown.
4. Simple weighted majority
   We chose the feature class with the maximum weighted vote among the 15 nearest classes. For each class the weight was inversely proportional to its corresponding feature's distance from the test feature.

$$w_i = \frac{1}{d_i}$$

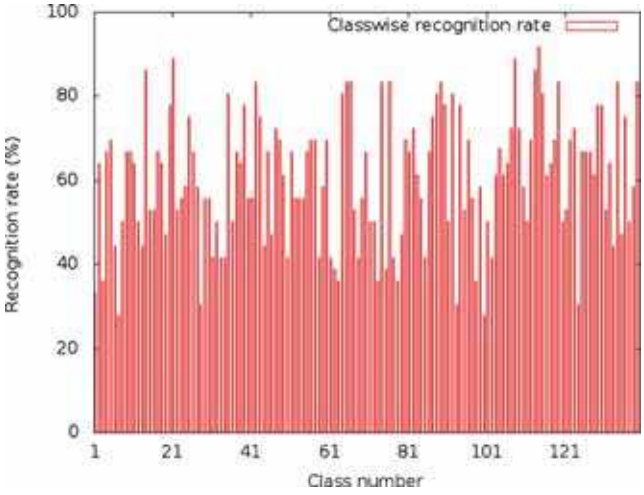


Figure 14. VFH Recognition rates on original dataset classwise (exponentially weighted recognition heuristic) ($a = \frac{1}{3}$).

where w_i is the weight of the ith feature's vote to its class W and d_i is its distance from the test feature.

5. Exponentially weighted majority:

   We found the feature class with the maximum exponential weight among the 15 nearest classes. A class W had the weight

$$w_i = e^{-a*d_i}$$

   where d_i is the distance of the ith feature among the 15 nearest features from the test feature and a is a constant (a was taken as $\frac{1}{3}$ in the final result of figure 15 as it gave the best recognition rate).

The results are shown in figure 15 and table 9. Exponentially weighted majority gives the best results. We downsampled (nearest neighbour interpolation) the original data by half to increase the speed of recognition and compared both. Downsampled dataset gives worse recognition rates with all the heuristics. This is expected, as downsampling (by half) makes contours and silhouttes hazy and ambiguous. This makes it difficult to represent key features and hence recognition rates fall down. However, there is a trade-off between speed and recognition. Downsampling increases speed but at a high cost of accuracy. Thus it is better not to downsample at all.

4.1d *Recognition using Speeded-Up Robust Feature (SURF)*: In SURF [48] the interest points are calculated by a Hessian matrix approach. The descriptor 'describes the distribution of the intensity content within the interest point neighbourhood, similar to the gradient information extracted by SIFT and its variants [48].' The descriptor is a 64 member vector for each interest point in the implementation used.

SURF is based on SIFT and uses optimizations, notably integral images, to speed up the process of computing the scale space. Instead of convolving with Gaussian kernels only, SURF uses the determinant of Hessian to guage interest points.

$$\mathcal{H}(x,\sigma) = \begin{bmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{bmatrix}$$

Here $L_{xx}(x,\sigma)$ is the convolution of the image with the second derivative of the Gaussian with standard deviation σ. In order to calculate the Hessian matrix faster, SURF uses

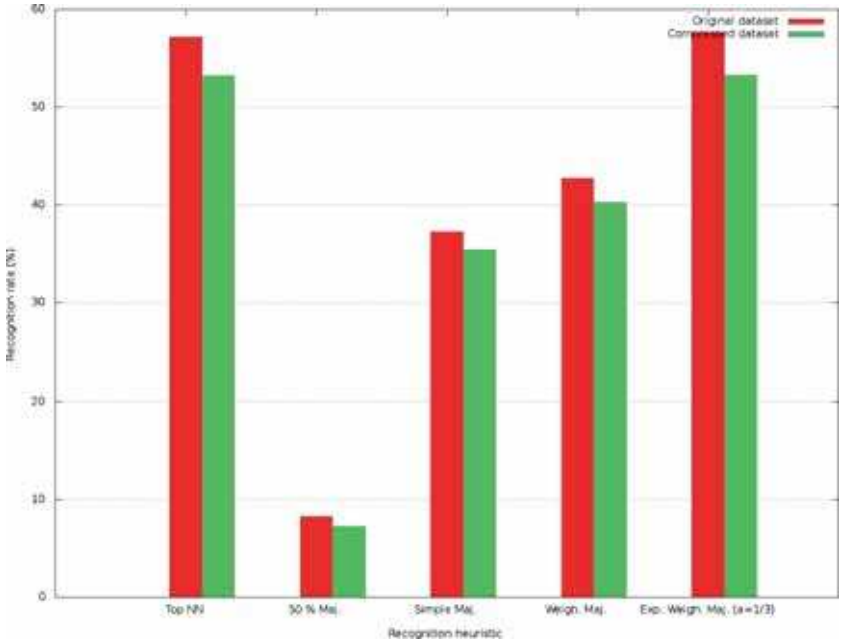


Figure 15. VFH recognition rates on original and downsampled (by 50%) datasets. From left: top nearest neighbour, 50% majority, simple majority, weighted majority and exponentially weighted majority.

Table 9. Average recognition rates for VFH. For exponentially weighted majority heuristic, $a = \frac{1}{3}$ was chosen for best results.

Category	Average recognition rate (%)	
	Original dataset	Downsampled dataset
Simple majority	37.2942	35.4692
Exponentially weighted majority	57.6671	53.2831
Simple weighted majority	42.7693	40.3293
Nearest neighbour	57.191	53.2434

approximations of these second order kernels in the form of box filters [49], calculating their convolutions by integral images [50]. An integral image I' of an image I is an image where each point $I'(x, y)$ stores the sum of all pixels in the rectangular area between the origin and $I(x, y)$. Usage of integral images ensures calculation of these convolutions in four lookups of the integral image. Because of the advantage offered in this process, there is no need to subsample the image in subsequent octaves, as was the case in SIFT.

Recognition: A procedure identical to that followed in Section 4.1b was followed. Multiple iterations (17) were made and test image features were matched with training image features via indexing in k-d trees. Identical recognition heuristics were followed. The results are summarised in table 10.

The results show that nearest neighbours heuristic gives the best recognition results with this feature.

4.1e *Recognition using neural networks*: As we can see from the previous section (for k-d trees), the best performer (in speed and accuracy) among VFH, SIFT and SURF for this dataset was VFH with exponentially weighted majority heuristic. In order to show that nearest neighbour using k-d trees based matching is better, we also tested our dataset using Neural Networks with VFH features.

We used a two-layer feed forward network, with hidden and output neurons as sigmoid units. The network was trained using scaled conjugate gradient backpropagation technique. We divided the entire dataset into three parts randomly:

1. Training set (T %)

Table 10. Average recognition rates for SURF. For exponentially weighted majority heuristic, $a = 1$ was chosen for best results.

Heuristic	Recognition rate (%)
Simple majority	46.60
Exponentially weighted majority	52.79
Simple weighted majority	52.77
Nearest neighbour	55.52

Table 11. VFH recognition rates for Neural networks on original dataset. T = 80, V = 15, X = 5.

No. of classes	No. of hidden layer neurons	No. of trials	Avg. classification rate (%)
140	10	10	1.16
	100	10	15.0
	150	10	13.4
	200	11	17.97
	300	10	18.46
	400	11	8.81
	500	11	16.24
	600	11	12.63

Table 12. VFH recognition rates for Neural networks on reduced dataset. T = 90, V = 5, X = 5.

No. of classes	No. of hidden layer neurons	No. of trials	Avg. classification rate (%)
2	10	32	90.62
	50	36	89.59
	100	36	86.12
	200	32	88.29
	300	35	76.429
	400	36	**90.973**
	500	32	83.59
	600	27	89.81
10	10	33	26.10
	100	24	55.56
	200	24	**59.26**
	300	30	47.04
	400	24	47.92
	500	20	39.727
	600	24	43.987
20	10	10	11.12
	100	12	**37.27**
	150	20	36.26
	200	19	27.35
	300	23	21.86
	400	26	22.41
	500	18	22.53
	600	20	25.83

Table 13. Regrouping scheme for dataset disambiguation. Classes not mentioned in this table were also included in the disambiguated dataset, reducing the overall size to 110 classes.

Group	Class numbers in this group
M1	1, 34, 101, 105, 124, 131, 125
M2	13, 118
M3	7, 70
M4	129, 126, 63
M5	135, 12, 83
M6	6, 82
M7	27, 30
M8	96, 103
M9	18, 57
M10	89, 120
M11	25, 104
M12	3, 134
M13	44, 88
M14	85, 116
M15	22, 36
M16	24, 29
M17	95, 97
M18	115, 66
M19	98, 99
M20	73, 79
M21	48, 137
M22	136, 45, 42

The network adjusts itself to the samples from this set, minimising the error.

2. Validation set (V %)

   This set is used to measure how generalised the network's classification rates are. When the generalisation stops improving, the training is stopped.

3. Test set (X %)

   These are used after the training phase to test the network's performance independently.

Table 14. VFH (exponentially weighted majority heuristic) recognition rates for disambiguated dataset (110 classes).

Category	Recognition rate (%)
Original dataset	60.3848
Downsampled dataset	55.7032

Table 11 shows the results achieved for different number of neurons in the hidden layer.

Table 17. Average recognition rates for feature combination on dataset categories for original dataset (140 classes). The strategy used was a combination of SIFT Weighted Majority, SURF Nearest Neighbours, VFH Nearest Neighbours with SURF prevailing in the case of a deadlock.

Category	Average recognition rate (%)
A	63.78
B1	58.9
B2	63.12
C	60.06
D	63.39

Table 15. Average recognition rates for feature combination. Each case has 3 heuristic names used with their corresponding feature names and a string 'X prevails' to indicate which feature is used in case of a deadlock like the case 4 of the list in Section 4.2b.

S. No.	Combination strategy		Average recognition rate (%)
1.	SIFT WM, SURF WM, VFH EW	VFH prevails	58.07
2.	SIFT SM, SURF SM, VFH EW	SURF prevails	60.54
3.	SIFT WM, SURF WM, VFH EW	SIFT prevails	64.50
4.	SIFT WM, SURF WM, VFH EW	SURF prevails	66.22
5.	SIFT WM, SURF NN, VFH EW	SURF prevails	68.11
6.	SIFT WM, SURF NN, VFH NN	SURF prevails	68.25

WM, Simple weighted majority; EW, Exponentially weighted majority; SM, Simple majority; NN, Nearest neighbour heuristic.

Table 16. Average categorywise recognition rates for all heuristics on original dataset.

Recog. heuristic	Avg. category-wise recognition rates (%)				
	A	C	B1	B2	D
1	54.39	**72.07**	47.61	53.77	51.57
2	63.71	60.36	58.60	62.69	62.99
3	**63.78**	60.06	58.90	**63.12**	**63.38**
4	60.11	57.35	58.30	61.84	62.20
5	60.58	63.06	55.57	60.36	59.44
6	52.13	50.75	57.31	57.81	53.54
SIFT EW	49.93	43.84	48.14	52.49	50.98
SIFT NN	50.13	43.24	47.68	51.85	50.19
SURF EW	55.32	60.06	48.29	53.77	52.16
SURF NN	58.58	54.95	52.38	57.59	57.48
VFH EW	53.40	48.61	**60.92**	62.06	54.34
VFH NN	53.34	48.33	60.06	61.73	54.68

Legend: 1: SIFT SM, SURF SM, VFH EW, SURF prevails, 2: SIFT WM, SURF NN, VFH EW, SURF prevails, 3: SIFT WM, SURF NN, VFH NN, SURF prevails, 4: SIFT WM, SURF WM, VFH EW, SIFT prevails, 5: SIFT WM, SURF WM, VFH EW, SURF prevails, 6: SIFT WM, SURF WM, VFH EW, VFH prevails,

WM, Simple weighted majority; EW, Exponentially weighted majority; SM, Simple majority; NN, Nearest neighbour heuristic. 'X prevails' indicates which feature is used in case of a deadlock. Bold numbers indicate the highest recognition rates in the respective categories A, B1, B2, C and D.

Since the results were very poor, we reduced the number of classes to investigate if it was the cause. Table 12 shows the results. We can see that the best recognition rates for 2, 10 and 20 classes were 90.973%, 59.26% and 37.27% respectively. When the number of classes was increased, the rate fell down. This decrease in the recognition rate shows that the number of classes in this dataset is huge compared to the images available per class. The

Table 18. Confusion matrix for the recognition approach SIFT WM, SURF NN, VFH NN, SURF prevails. Top 5 confused classes are given with each class (parenthesis have the percentage of confusion) for this recognition approach.

Classes	SIFT WM, SURF NN, VFH NN, SURF prevails					
11	11(94.12)	10(5.88)	–	–	–	–
14	14(60.61)	139(15.15)	32(3.03)	37(3.03)	65(3.03)	69(3.03)
16	16(93.75)	81(3.13)	83(3.13)	–	–	–
25	25(36.36)	104(30.3)	9(3.03)	52(3.03)	53(3.03)	67(3.03)
27	27(65.63)	30(25)	32(3.13)	114(3.13)	119(3.13)	–
30	30(50)	27(28.13)	119(15.63)	32(3.13)	136(3.13)	–
31	31(100)	–	–	–	–	–
34	34(40.63)	105(18.75)	119(6.25)	136(6.25)	30(3.13)	48(3.13)
35	35(93.75)	11(3.13)	31(3.13)	–	–	–
37	37(45.16)	130(6.45)	4(3.23)	34(3.23)	50(3.23)	51(3.23)
38	38(93.94)	80(3.03)	121(3.03)	–	–	–
42	42(42.42)	45(9.09)	127(6.06)	2(3.03)	5(3.03)	8(3.03)
43	43(83.87)	26(3.23)	48(3.23)	53(3.23)	72(3.23)	109(3.23)
47	47(80.95)	25(4.76)	31(4.76)	87(4.76)	121(4.76)	–
55	55(77.78)	47(3.7)	53(3.7)	54(3.7)	72(3.7)	77(3.7)
49	49(36.67)	77(13.33)	99(6.67)	118(6.67)	122(6.67)	137(6.67)
56	56(53.13)	35(18.75)	11(6.25)	140(6.25)	12(3.13)	38(3.13)
59	59(51.52)	139(6.06)	45(3.03)	60(3.03)	62(3.03)	63(3.03)
61	61(96.77)	83(3.23)	–	–	–	–
63	63(53.57)	128(17.86)	126(14.29)	129(7.14)	2(3.57)	59(3.57)
64	64(54.55)	70(18.18)	7(12.12)	1(3.03)	47(3.03)	69(3.03)
70	70(69.7)	64(12.12)	7(6.06)	47(3.03)	77(3.03)	88(3.03)
76	76(27.27)	79(6.06)	110(6.06)	132(6.06)	10(3.03)	18(3.03)
71	71(37.5)	107(8.33)	115(8.33)	22(4.17)	41(4.17)	48(4.17)
77	77(51.72)	41(3.45)	46(3.45)	61(3.45)	71(3.45)	72(3.45)
79	79(27.27)	121(6.06)	130(6.06)	140(6.06)	1(3.03)	30(3.03)
80	80(35.48)	122(6.45)	11(3.23)	18(3.23)	22(3.23)	35(3.23)
85	85(69.7)	116(9.09)	120(9.09)	19(3.03)	34(3.03)	56(3.03)
86	86(96.88)	128(3.13)	–	–	–	–
93	93(85.71)	97(4.76)	121(4.76)	122(4.76)	–	–
94	94(43.75)	97(9.38)	80(6.25)	17(3.13)	34(3.13)	35(3.13)
95	95(37.5)	139(12.5)	6(6.25)	17(6.25)	29(6.25)	76(6.25)
98	98(17.24)	71(10.34)	83(6.9)	99(6.9)	114(6.9)	133(6.9)
101	101(76.47)	47(5.88)	44(2.94)	54(2.94)	67(2.94)	70(2.94)
102	102(48.28)	40(13.79)	91(6.9)	22(3.45)	53(3.45)	67(3.45)
112	112(25.81)	121(12.9)	134(6.45)	137(6.45)	16(3.23)	37(3.23)
117	117(100)	–	–	–	–	–
121	121(39.39)	88(6.06)	19(3.03)	48(3.03)	49(3.03)	53(3.03)
122	122(41.94)	80(9.68)	10(6.45)	5(3.23)	11(3.23)	15(3.23)
125	125(76.47)	55(8.82)	74(2.94)	101(2.94)	109(2.94)	120(2.94)
126	126(73.53)	129(17.65)	2(2.94)	4(2.94)	128(2.94)	–
129	129(45.45)	126(33.33)	128(12.12)	63(6.06)	2(3.03)	–
130	130(81.82)	44(3.03)	66(3.03)	67(3.03)	88(3.03)	119(3.03)
134	134(67.65)	3(5.88)	126(5.88)	4(2.94)	18(2.94)	44(2.94)
135	135(72.73)	83(27.27)	–	–	–	–
136	136(90.91)	85(3.03)	105(3.03)	114(3.03)	–	–
137	137(94.44)	117(5.56)	–	–	–	–
139	139(90.32)	14(3.23)	45(3.23)	72(3.23)	–	–
140	140(100)	–	–	–	–	–

number of images available in this dataset per class is not enough for a Neural network to correctly classify an image into 140 classes. Thus, for the current dataset, Neural Networks are not a good choice as a classifier.

4.2 *Final results*

Even for the VFH and Nearest Neighbour (k-d tree based) classifier, the results are not very good. This is partly due to the reason that many signs of the ISL Dataset have very similar gestures (see figure 7).

4.2a *Merging ambiguous classes*: The dataset was further disambiguated on the basis of shape similarity by merging some classes. One hundred and forty classes were regrouped into 110 classes and the VFH features were reapplied. Regrouping scheme is shown in table 13. The results are shown in tables 13 and 14.

4.2b *Combining VFH, SIFT and SURF*: It was thought that a stronger and better prediction should be made if we make use of all of the above features under a certain heuristic.

We used a general strategy of classifier combination based on majority voting as explained below:

1. Extract VFH, SIFT and SURF features. Calculate individual class predictions for weighted majority heuristic for SIFT, nearest neighbour majority heuristic for SURF and nearest neighbour heuristic for VFH.
2. If the labels predicted by the three heuristics are identical, then predict that label as the final output label.
3. If the labels predicted by any two heuristics are identical, then predict that label as the final output label.
4. If the labels predicted by the three heuristics are all different, then predict the label given by one of the features (SURF/SIFT/VFH) as the final output label.

We experimented with a lot of strategies as given in table 15 which shows the results calculated by averaging on 17 test trials. The categorywise recognition rates of these strategies are given in tables 16 and 17. The combination strategy, SIFT WM, SURF NN, VFH NN and SURF prevails, of item 6 in table 15 gave the best results. Confusion matrix for this strategy are given in table 18. This strategy (SIFT WM, SURF NN, VFH NN and SURF prevails) was also applied on categories (A, $B1$, $B2$, C and D) of our dataset without disambiguation. The results of this application are shown in table 17. In the fingerspelling category A,

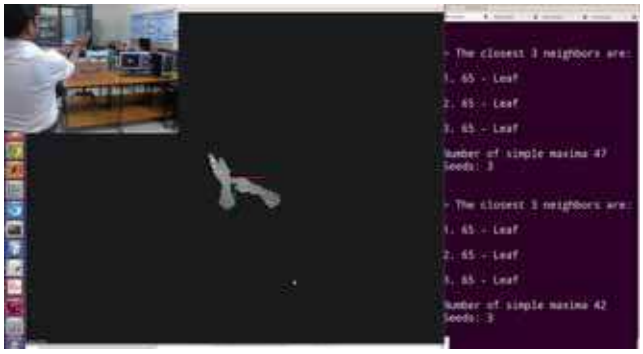


Figure 17. A real-time system snapshot. Left panel: Clustered hands. Right panel: Recognition predictions. *Inset*: Gesture for the sign 'Leaf' [51].

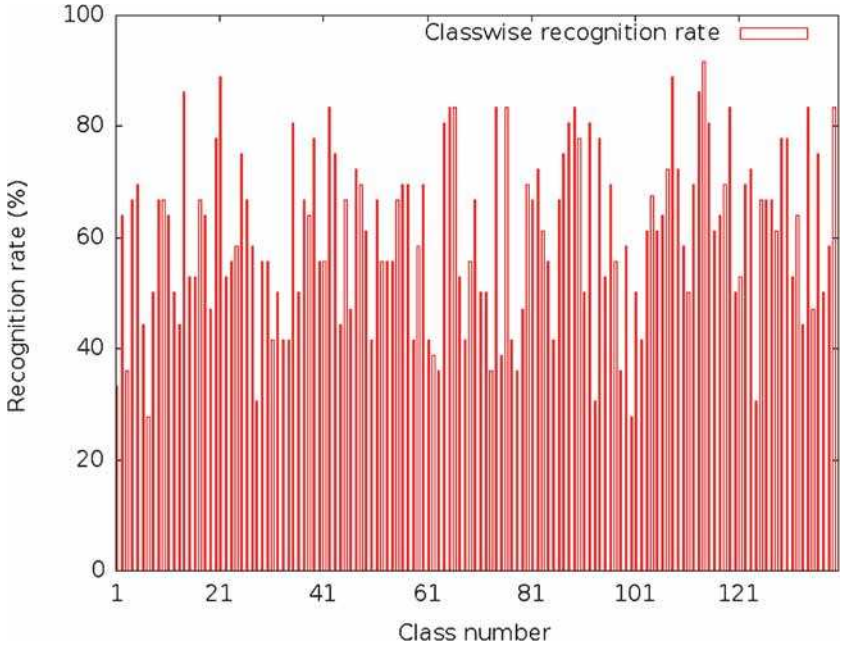


Figure 16. Classwise recognition rates of fingerspelling category (A).

we achieved above 90% recognition rates for 13 signs (11, 16, 31, 35, 38, 61, 86, 107, 117, 136, 137, 139 and 140) and 100% recognition for signs 31, 117 and 140. Overall 16 distinct alphabets (A, B, D, E, F, G, H, K, P, R, T, U, W, X, Y, Z) were recognised with an average accuracy rate of 90.68% (classes 11, 16, 31, 35, 38, 43, 55, 61, 86, 101, 117, 125, 130, 136, 137, 140). Classwise recognition rate for this category (A) is shown in figure 16. A snapshot of the system in use is shown in figure 17. Representative images for the signs in the ISL dataset are shown in figure 18.

5. Conclusion

Gesture recognition is a complex and challenging task. We designed a software stack which could be easily used and

Figure 18. The ISL Dataset has 140 signs. Representative images of each sign are given here (see table 5 for symbol names).

extended. There is a dearth of Kinect depth datasets for Indian Sign Language signs (figure 18). ISL has hardly received any attention from the research community compared to ASL and other such sign language systems worldwide. We shall release all the data and code under an open-source license for the research community. In the fingerspelling category of our dataset, we achieved above 90% recognition rates for 13 signs and 100% recognition for three signs with overall 16 distinct alphabets (A, B, D, E, F, G, H, K, P, R, T, U, W, X, Y, Z) recognised with an average accuracy rate of 90.68%. Many approaches were tried out in this regard and would constitute a useful platform to engage in the future [52]. Future directions to further develop this system are as follows:

1. In the current work, we tried to tackle only static signs. But in majority of the communication through sign languages, much of the semantic content lies in movement, facial expressions and stress placed on the sign. Thus there is scope for development of a complete system which could be used as an accompaniment for the disabled to communicate, in which all elements of a sign language would be used.
2. Language and context models can be applied to generate complete sentences. This might be done using HMMs.
3. Gesture recognition could also be used in recognition of Cued Speech – a sign system in which both lipreading and hand gestures are used in conjunction.
4. This system could be ported to run on a portable embedded system that could be taken by a disabled person and used anywhere for interpretation.

References

[1] Schmitz M, Endres C and Butz A 2008 A survey of human-computer interaction design in science fiction movies. In: *Proceedings of the 2nd international conference on INtelligent TEchnologies for interactive enterTAINment*, page 7. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering)

[2] Bauman D 2008 *Open your eyes: Deaf studies talking*. University of Minnesota Press

[3] Muni B 1951 *Natya Shastra.* Calcutta: Asiatic Society of Bengal

[4] Bulwer J 1648 *Philocopus, or the deaf and dumb man's friend*. London: Humphrey and Moseley

[5] Cornett R O 1967 *Cued speech*. University of Nebraska Media Center. Captioned Films for the Deaf

[6] Geetha M and Manjusha U 2012 A vision based recognition of indian sign language alphabets and numerals using B-spline approximation. *Int. J. Comp. Sci. Eng. (IJCSE)*

[7] Pugeault N and Bowden R 2011 Spelling it out: Real-time ASL fingerspelling recognition. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1114–1119

[8] ISL 2011 Sign dictionaries for indian deaf and dumb population. Sign Language Unit, The Faculty of Disability Management & Special Education (FDMSE) of Ramakrishna Mission Vidyalaya, Perianaickenpalayam, Coimbatore, India. indiansignlanguage.org

[9] Mitra S and Acharya T 2007 Gesture recognition: A survey. *IEEE Trans. Syst. Man Cybern. Part C: Appl. Rev.* 37(3): 311–324

[10] Bilal S, Akmeliawati R, El Salami M J and Shafie A A 2011 Vision-based hand posture detection and recognition for sign language–a study. In: *4th International Conference on Mechatronics (ICOM), 2011*, pages 1–6

[11] Quek F K and Zhao M 1996 Inductive learning in hand pose recognition. In: *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition, 1996*, pages 78–83

[12] Starner T, Weaver J and Pentland A 1998 Real-time american sign language recognition using desk and wearable computer based video. *IEEE Trans. Pattern Anal. Mach. Intell.* 20(12): 1371–1375

[13] Luis-Pérez F E, Trujillo-Romero F and Martínez-Velazco W 2011 Control of a service robot using the mexican sign language. In: *Adv. Soft Comput.*, pages 419–430. Springer

[14] Rekha J, Bhattacharya J and Majumder S 2011 Shape, texture and local movement hand gesture features for Indian Sign Language recognition. In: *3rd International Conference on Trendz in Information Sciences and Computing (TISC), 2011*, pages 30–35

[15] Singha J and Das K 2013 Indian sign language recognition using eigen value weighted euclidean distance based classification technique. *arXiv preprint* arXiv:1303.0634

[16] Bhuyan M, Kar M K and Neog D R 2011 Hand pose identification from monocular image for sign language recognition. In: *2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, pages 378–383

[17] Kenn H, Megen F V and Sugar R 2007 A glove-based gesture interface for wearable computing applications. In: *4th International Forum on Applied Wearable Computing (IFAWC), 2007*, pages 1–10

[18] Hernandez-Rebollar J L, Lindeman R W and Kyriakopoulos N 2002 A multi-class pattern recognition system for practical finger spelling translation. In: *Proceedings of the 4th IEEE International Conference on Multimodal Interfaces*, ICMI '02, pages 185–, Washington, DC, USA. IEEE Computer Society

[19] Liang R-H and Ouhyoung M 1998 A real-time continuous gesture recognition system for sign language. In: *Proceedings of the Third IEEE international conference on automatic face and gesture recognition*, 1998, pages 558–567

[20] Saengsri S, Niennattrakul V and Ratanamahatana C 2012 Tfrs: Thai finger-spelling sign language recognition system. In: *Second International Conference on Digital Information and Communication Technology and it's Applications (DICTAP), 2012*, pages 457–462

[21] Van den Bergh M and Van Gool L 2011 Combining RGB and ToF cameras for real-time 3D hand gesture interaction. In: *IEEE Workshop on Applications of Computer Vision (WACV)*, 2011, pages 66–72

[22] Wang F and Zhang C 2007 Feature extraction by maximizing the average neighbourhood margin. In: *IEEE Conference on Computer Vision and Pattern Recognition, 2007. CVPR'07*, pages 1–8

[23] Argyros A and Lourakis M I A 2006 Binocular hand tracking and reconstruction based on 2D shape matching. In: *18th International Conference on pattern recognition, 2006. ICPR 2006*. volume 1, pages 207–210

[24] Keskin C, Kıraç F, Kara Y E and Akarun L 2013 Real time hand pose estimation using depth sensors. In: *Consumer depth cameras for computer vision*, pages 119–137. Springer

[25] Ren Z, Yuan J and Zhang Z 2011 Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera. In: *Proceedings of the 19th ACM international conference on Multimedia*, MM '11, pages 1093–1096, New York, NY, USA

[26] Shotton J, Sharp T, Kipman A, Fitzgibbon A, Finocchio M, Blake A, Cook M and Moore R 2013 Real-time human pose recognition in parts from single depth images. *Commun. ACM* 56(1): 116–124

[27] Kramer J, Parker M, Herrera D, Burrus N and Echtler F 2012 *Hacking the kinect*. Apress

[28] ElKoura G and Singh K 2003 Handrix: Animating the human hand. In: *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on computer animation*, pages 110–119. Eurographics Association

[29] PrimeSense 2011 OpenNI platform 1.0

[30] Rusu R B and Cousins S 2011 3D is here: Point cloud library (PCL). In: *IEEE International Conference on Robotics and Automation (ICRA), 2011*, pages 1–4

[31] Buades A, Coll B and Morel J M 2005 A non-local algorithm for image denoising. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005*, vol. 2, pages 60–65

[32] Tukey J W 1977 Exploratory data analysis. Reading, MA, 231

[33] Shi J and Malik J 2000 Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* 22(8): 888–905

[34] Lowe D G 2004 Distinctive image features from scale-invariant keypoints. *Int. J. Comp. Vis.* 60(2): 91–110

[35] Lindeberg T 2012 Scale invariant feature transform. *Scholarpedia* 7(5): 10491

[36] Lindeberg T 1994 Scale-space theory: A basic tool for analyzing structures at different scales. *J. Appl. Stat.* 21(1–2): 225–270

[37] Lindeberg T 1998 Feature detection with automatic scale selection. *Int. J. Comp. Vis.* 30(2): 79–116

[38] Bundy A and Wallen L 1984 Difference of gaussians. In: Bundy A and Wallen L (eds) *Catalogue of artificial intelligence tools*, symbolic computation, page 30. Springer, Berlin Heidelberg

[39] Bhatnagar S 2007 Adaptive newton-based multivariate smoothed functional algorithms for simulation optimization. *ACM Trans. Model. Comput. Simul.* 18(1): 2:1–2:35

[40] Lindeberg T 1993 *Scale-space theory in computer vision*. Springer

[41] Weickert J, Ishikawa S and Imiya A 1999 Linear scale-space has first been proposed in japan. *J. Math. Imaging Vis.* 10(3): 237–252

[42] Lindeberg T 2013 Generalized axiomatic scale-space theory. *Adv. Imaging Electron Phys.* 178: 1

[43] Yuille A L and Poggio T A 1986 Scaling theorems for zero crossings. *IEEE Trans. Pattern Anal. Mach. Intell.* 8(1): 15–25

[44] Babaud J, Witkin A P, Baudin M and Duda R O 1986 Uniqueness of the gaussian kernel for scale-space filtering. *IEEE Trans. Pattern Anal. Mach. Intell.* 8(1): 26–33

[45] Allaire S, Kim J J, Breen S L, Jaffray D A and Pekar V 2008 Full orientation invariance and improved feature selectivity of 3d SIFT with application to medical image analysis. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08*, pages 1–8

[46] Rusu R B, Bradski G, Thibaux R and Hsu J 2010 Fast 3D recognition and pose using the viewpoint feature histogram. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2010*, pages 2155–2162

[47] Rusu R B, Blodow N and Beetz M 2009 Fast point feature histograms (FPFH) for 3d registration. In: *IEEE International Conference on Robotics and Automation, 2009. ICRA'09*, pages 3212–3217

[48] Bay H, Tuytelaars T and Van Gool L 2006 Surf: Speeded up robust features. In: *Computer Vision–ECCV 2006*, pages 404–417. Springer

[49] McDonnell M 1981 Box-filtering techniques. *Comp. Graph. Image Process.* 17(1): 65–70

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

[51] Ansari Z 2013a Gesture recognition for Indian Sign Language with Kinect Xbox 360. https://www.youtube.com/watch?v=2oqD-_UCHxQ. Accessed: 2013-06-18

[52] Ansari Z A 2013b *Gesture recognition for Indian sign language*. Master's thesis, Indian Institute of Technology Jodhpur, India