Spelling It Out: Real–Time ASL Fingerspelling Recognition

Nicolas Pugeault & Richard Bowden
Centre for Vision, Speech and Signal Processing
University of Surrey
{n.pugeault,r.bowden}@surrey.ac.uk
http://personal.ee.surrey.ac.uk/Personal/N.Pugeault
http://personal.ee.surrey.ac.uk/Personal/R.Bowden

Abstract

This article presents an interactive hand shape recognition user interface for American Sign Language (ASL) finger-spelling. The system makes use of a Microsoft Kinect device to collect appearance and depth images, and of the OpenNI+NITE framework for hand detection and tracking. Hand-shapes corresponding to letters of the alphabet are characterized using appearance and depth images and classified using random forests. We compare classification using appearance and depth images, and show a combination of both lead to best results, and validate on a dataset of four different users.

This hand shape detection works in real-time and is integrated in an interactive user interface allowing the signer to select between ambiguous detections and integrated with an English dictionary for efficient writing.

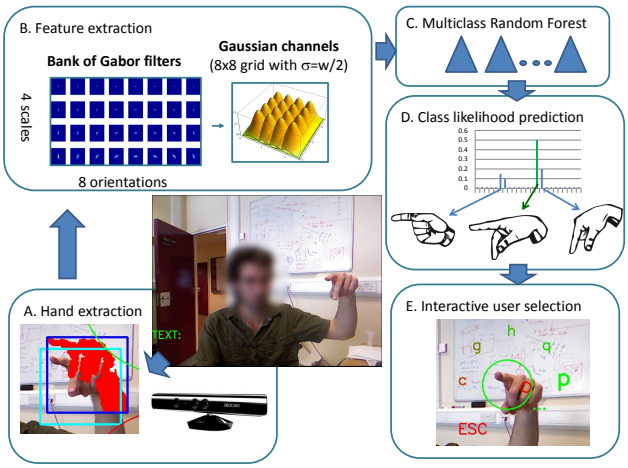


Figure 1. Illustration of the interactive ASL finger-spelling system.

1. Introduction

This article presents an interactive finger-spelling graphical user interface based on American Sign Language (ASL) using a Microsoft Kinect device [8]. The proposed system is capable of classifying finger-spelling hand shapes in real time (on an off the shelf laptop computer), and overcomes classification ambiguity by offering an easy way for the user to select between ambiguous choices. We believe this offers a novel interface for keyboard-less interaction and for sign language learning.

Sign language recognition is a very hard problem, and despite recent progresses automatic sign language recognition systems are still in their infancy (see, eg, [11, 9]). Typically, signs are characterized not only hand shapes, but also hand movements and other non-manual features such as facial expressions and body posture.

This work focuses on the much simpler problem of recognizing a limited set of hand shapes, namely the ones forming the ASL finger-spelling alphabet—as shown in Fig. 2. Because American finger-spelling is single-handed, this removes the difficulties of hands occluding one another. Also, the signs we consider are all static, allowing us to focus on hand-shape per se, rather than motion. Despite being a simplified problem, hand shape recognition remains a challenging one. First, some of the signs in the alphabet are visually very similar, for example the letters *a, e, m, n, s* and *t* are all signed with a fist-like hand shape, and only distinguished by the position of the thumb—as illustrated in Fig. 4. The thumb position itself is barely visible for *m* and *t*, making decision difficult. Second, there is a large amount of variation in how the sign is done by different persons, and the hand shape appearance will vary widely depending on the hand's pose—this is illustrated for the letter *p* in Fig. 3.

This article was published in the proceedings of the *1st IEEE Workshop on Consumer Depth Cameras for Computer Vision*, in conjunction with *ICCV'2011*, Barcelona, Spain.

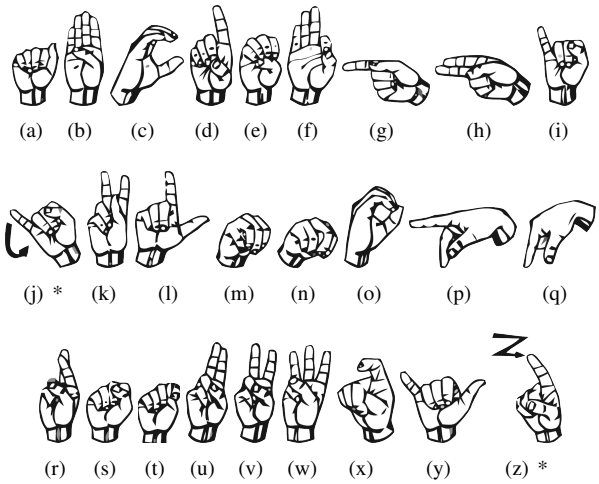


Figure 2. ASL finger spelling alphabet. We will ignore the letters *j* and *z* as they involve motion, and rely on dictionary lookup for those. Reproduced from [15]

Third, the differences between people's hands and natural dexterity leads to differences in the execution of signs between different signers. Finally, for a finger-spelling user interface to be usable in practice, it needs to run in real-time (> 10Hz) on commonly available computers.

The most similar previous work is probably the article by Isaac & Foo [6], who proposed an ASL finger-spelling recognition system based on neural networks applied to wavelets features. As in this article, they focus on static hand shapes, and report a recognition rate of 99% but do not specify the size of the dataset and number of different subjects. Bergh & Van Gool [14] propose a method that recognises 6 hand poses from a single user. Their system is based on a concatenation of depth and color-segmented images, and the recognition uses a combination of Haar wavelets and neural network. In contrast to this work, we consider 24 different hand–poses provided by ASL, some of them very similar, across 5 different users. Liwicki & Everingham [7] attempted British Sign Language (BSL) finger-spelling recognition using a combination of Histogram of Gradients (HOG) features and Hidden Markov Models (HMMs). This is intrinsically different to the present work as BSL is two handed whereas ASL is single-handed. Ong & Bowden [10] proposed a hand shape classifier based on boosted classifiers and shape context features, claiming 97.4% recognition rate on 300 artificial hand shapes.

This article differs from these previous works in several respects. First we will consider features extracted not only from standard intensity images, but also from depth images provided by a Microsoft Kinect device. Depth information is robust to illumination and skin pigment differences, offering a more reliable detection. Second, we propose a real-time approach that can recognize 24 of the 26 signs with

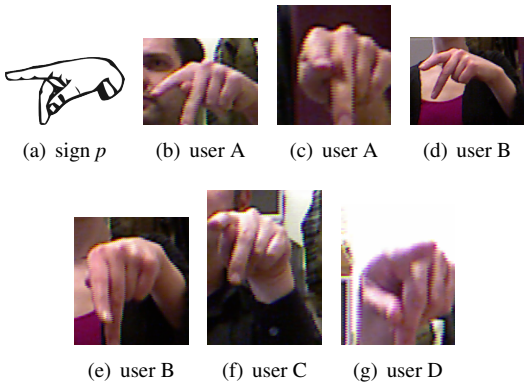


Figure 3. Illustration of the variability in the dataset, due to small pose variations, and different users finger-spelling styles. All images show the bounding boxes extracted for the color image for the letter *p*.

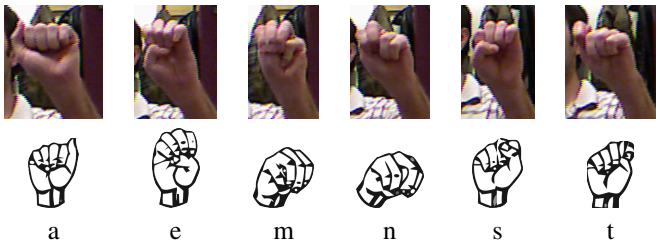


Figure 4. Illustration of the ambiguity in the dataset between different classes. All are represented by a closed fist, and differ only by the thumb position, leading to higher confusion levels. All examples are taken from the same user.

very good accuracy across different users. Finally, we embedded our finger-spelling detection into an interactive user interface that overcomes the ambiguity between signs by proposing to the user an easy way to choose between plausible hypotheses, and interfacing signing with a dictionary lookup.

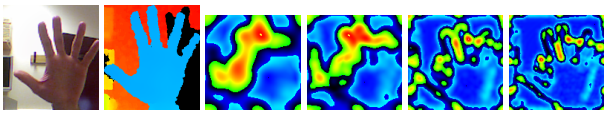
In the following, we start by describing the hand-shape recognition approach (section 2), before discussing the dataset and performance (section 3), and then presenting the user interface (section 4).

2. Methods

In this section we describe how the ASL finger-spelling hand shapes are learnt and recognized by the system. First, section 2.1 describes the hand extraction and tracking, then section 2.2 exposes the extracted hand features and finally, section 2.3 discusses the classification method. This is illustrated in Fig. 1.

2.1. Hand Extraction & Tracking

The detection and tracking of the user's hand is depth-based, and performed using the OpenNI+NITE [2, 12] framework on a Microsoft Kinect front–end [8]. This li-



(a) color (b) depth (c) scale 0 (d) scale 1 (e) scale 2 (f) scale 3

Figure 5. Illustration of the feature extraction process. a) color patch extracted around the hand; b) depth patch extracted around the hand; c-f) responses of the Gabor convolution for each scale, cumulated over all orientations.

brary provides functions for detecting hands from specified gestures (eg, 'wave'), and for tracking detecting hands in 3D space (using the depth image provided by the Kinect). This tracking only provides a position in space, the hand is then segmented from the depth image assuming that the hand is a continuous region which depth varies no more than 20 cm. in space. These segmented pixels were then used to form bounding boxes around the hand, both in the depth and color camera image—see Fig. 1A.

Moreover, because we only consider static signs, we discard frames where the hand is moving. This insures that we do not attempt to classify the transition between two handshapes. In practice, we only consider frames where the hand velocity is lower than 0.01 and where the depth patch variance over the last $N = 10$ frames is lower than 0.02—those values were set experimentally for convenience of use, and can be tuned in a user-specific manner (eg, a more competent finger-speller may want to reduce the time window).

2.2. Hand shape features

The hand shape features we use are based on Gabor filtering of the intensity and depth images—see Fig. 1B. Once the hand is detected, both depth and intensity bounding boxes are resized to 128×128, before being convolved with a bank of Gabor filters at 4 scales and 4 orientations [5]. The reason for using a bank of Gabor filters is that they are apt at capturing the contrast (in depth or intensity) of the image patch at different scales, and therefore capturing at the same time the overall shape of the hand and the details of the fingers (see, *e.g.*, Fig. 5c-f, for a depth patch). A 2D Gabor filter is typically formed by the formula:

$$g(x, y, \lambda, \sigma, \theta) = \exp\left(\frac{x'^2 + y'^2}{2\sigma^2}\right) \exp\left(i\left(\frac{2\pi x}{\lambda}\right)\right), \quad (1)$$

where $x' = x\cos\theta + y\sin\theta$ and $y' = -x\sin\theta + y\cos\theta$, λ is the wavelength, θ is the orientation and σ is the variance of the Gaussian envelope. These can be computed real-time (30 Hz) for 128×128 images, on a typical laptop (Intel Core i5 M430 processor at 2.27GHz). The convolution results are arrays of floating point values of the same size as the image, written as $J_1 \ldots J_{16}$. This is illustrated in Fig. 5.

The filter responses are then averaged across overlapping Gaussian basis functions positioned on a regular 8×8 grid.

$$C(i, j, k) = \sum_{x,y} J_k(x, y) \exp - \frac{(x - x_i)^2 + (y - y_j)^2}{2s^2} \quad (2)$$

where i, j indicate the horizontal and vertical index of the cell ($x_i = 16i - 8$, and $y_j = 16j - 8$), k indicates the Gabor filter index, and $s = 8$ the width of the Gaussian filter. This forms a feature vector of dimension 1,024 ($= 8 \times 8 \times 4 \times 4$) for both depth and intensity information, and 2,048 for the joint feature vector.

2.3. Multi-class random forest classification

The learning and classification is performed using a multi-class random forest. Random forests were introduced by Amit & Geman [3] and later Breiman [4], and can be viewed as an extension of bagging to classification trees. Random forest have shown competitive performance in learning, can handle well large datasets and large feature space, and can be parallelized for fast training and recall performance. In this paper, we make use of a version similar to the one proposed by Breiman, detailed in the following.

A random forest consists of a population of randomized classification tree T_i, each trained with a random bootstrap sample $S_i \subset S^*$ of the original training data S^*. Each classification tree is built by recursively partitioning the input feature space at each node, so as to maximize the reduction in the entropy of the class distribution. Specifically, at each node a random subset of feature dimensions are selected, and over all these dimensions, the threshold leading to the largest reduction in class distribution entropy is chosen. The partition ends when all samples that fall in a node are of a single class, or when a maximal depth is reached. Then this node is labeled as the majority class in the training samples it contains. Given a new sample x, this tree will then predict the class $T_i(x)$ corresponding to the label of the leaf node it falls into.

Finally, the class predicted by all trees in the forest are then collated to form a class probability distribution estimate over the 24 target signs. Overall, the forest yields a confidence that the feature vector x indicates a hand shape c:

$$p[c] = \frac{1}{N} \sum_{i<N} \delta_c(T_i(x)), \quad (3)$$

where N is the number of trees in the forest, $T_i(x)$ is the leaf of the ith tree T_i into which x falls, and $\delta_c(a)$ is the Kronecker delta function ($\delta_c(a) = 1$ iff. $c = a$, $\delta_c(a) = 0$ otherwise). This probability distribution is then used to propose likely candidates to the user—see Fig. 1D.

3. Results

The dataset we used for this work contains over 500 samples of each sign, recorded from 4 different persons (non-

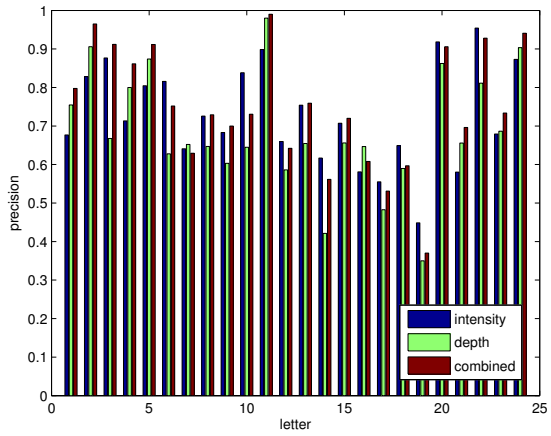


Figure 7. Comparison of the performance for all letters, using appearance information only, depth only, and a combination of both.

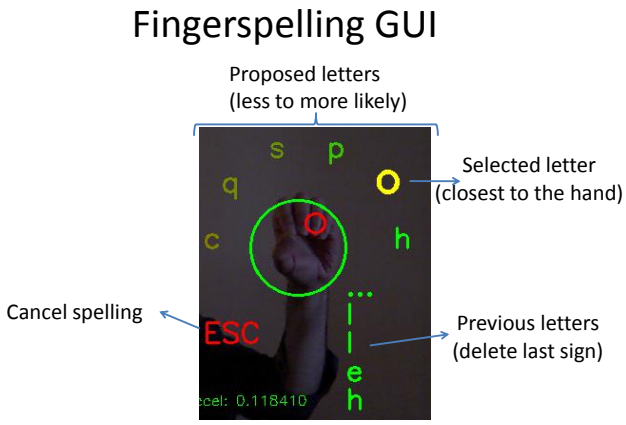


Figure 8. Illustration of the finger-spelling user interface. The letter selection interface: the user can select between plausible choices by moving his hand in this direction.

native to sign language), amounting to a total of 48,000 samples. Half of this data was kept for validation purpose and half was used for training the random forest. The subjects were asked to make the sign facing the Kinect device and to move their hand around while keeping the hand shape fixed, in order to collect a good variety of background and viewing angles—see Fig 6 for an illustration of the variety in size, background and orientation.

The overall performance is recorded in Fig. 7 when using appearance (intensity) only, depth only and a combined feature vector. The best performance is obtained using the combined vector (mean precision 75%), followed by appearance (mean precision 73%) and depth (mean precision 69%). The slightly lower performance of depth is compensated by a greater robustness to environmental circumstances, like lighting.

Table 1 records the confusion matrix for the detection of all letters, using a combined feature vector. This shows small confusion between similar looking hand-shapes, such as *r* and *u* (17%), and also *a, e, m* and *s*. The less certain class is the *t* with only 37% precision, and significant confusion with *s, n, e* and *a*. This is expected as these letters are signed as a fist that only differs by the thumb position, that is barely visible in some cases (depending on subject's physiognomy and dexterity). In the following we discuss a graphical user interface that addresses this confusion by offering to the user the possibility to choose select between ambiguous candidates as in Fig. 1E.

4. Interactive user interface

This hand shape recognition mechanism is embedded in an interactive finger-spelling user interface with a Microsoft Kinect front-end, that runs on a standard laptop under Microsoft Windows or Linux operating systems. The essential concept of this interface is to offer the user a chance to see non-maximal candidates when the classifier has low confidence in its prediction, and to select the correct letter from a list.

In practice, the hand is detected and tracked using a waving gesture and tracked using the OpenNI+NITE framework [2, 12]. Once the initial gesture is performed, the hand is tracked by the software, and finger-spelling can start.

Whenever the user's hand stops moving, the system enters finger-spelling mode. This is assessed by considering whether the tracked hand velocity is lower than $\alpha = 0.01$ (insuring that the hand is not moving), and the cumulated variance on the depth image over a time window of $N = 10$ is lower than $\beta = 0.02$ (insuring that the user is not moving his fingers). These parameters can be tuned to offer fast interaction (ie, quick response to a change in hand shape) to a skilled finger-speller and simpler interaction for a beginner (ie, more stable predictions). When this is the case, the hand patch is processed and hand-shapes are predicted from the combined feature vector. The most likely hypotheses are then proposed to the user in a semi-circle above the user's hand, allowing for easy selection by a small move of the hand in the direction of the desired letter. Moreover, downward motions can be used to delete the last letter spelled (down-right) or cancel spelling and return to the general interface (down-left). This is illustrated in Fig. 8.

When not in spelling mode, the user can move his hand to different areas of the screen to perform certain actions—as illustrated in Fig. 9. The top-right corner of the screen displays the currently spelled word, and the user can validate the word by a wave towards this corner. It is then added to the currently spelled sentence at the bottom-left

Figure 6. Illustration of the variety of the dataset. This array shows one image from each user and from each letter, displayed with relative size preserved. The size, orientation and background can change to a large extent. The full dataset contains approximately 100 images per user per letter.

	a	b	c	d	e	f	g	h	i	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y
a	**0.75**		0.05									0.05		0.05					0.10					
b	0.03	**0.83**	0.03								0.03								0.07					
c			**0.57**	0.13	0.03	0.03	0.03		0.07		0.03								0.03					0.03
d				**0.37**		0.13	0.03		0.07	0.03	0.07						0.17	0.03					0.10	
e		0.07	0.03		**0.63**								0.03	0.03			0.03	0.10	0.07					
f		0.30	0.10		0.05	**0.35**					0.15							0.05						
g	0.05				0.05	0.05	**0.60**								0.20				0.05					
h							0.03	**0.80**		0.03			0.03		0.10									
i		0.03	0.03	0.03		0.03			**0.73**				0.03						0.03				0.03	
k			0.03	0.03		0.07	0.03			**0.43**	0.03		0.03				0.07				0.20		0.03	0.03
l				0.13							**0.87**													
m	0.10		0.03		0.10		0.03				0.03	**0.17**	0.10		0.03	0.03		0.27					0.07	
n	0.17	0.10		0.03					0.03			0.10	**0.23**	0.07				0.13	0.10		0.03			
o	0.10		0.30	0.13		0.03	0.07		0.03	0.07	0.03			**0.13**	0.07			0.03						
p			0.07	0.10		0.03		0.10	0.03						**0.57**	0.07			0.03					
q	0.03						0.07								0.07	**0.77**			0.03				0.03	
r			0.03	0.03	0.03	0.07			0.03								**0.63**			0.13	0.03			
s	0.30		0.13	0.03	0.07				0.13				0.03	0.03				**0.17**	0.07				0.03	
t	0.33				0.13			0.03	0.03			0.07	0.03	0.10				0.20	**0.07**					
u		0.17		0.03													0.10			**0.67**		0.03		
v			0.03								0.03						0.03			0.03	**0.87**			
w			0.03			0.03								0.03							0.37	**0.53**		
x	0.03	0.03		0.17		0.07	0.07		0.20	0.03			0.07				0.10						**0.20**	0.03
y	0.07					0.07			0.10															**0.77**

Table 1. Confusion matrix of all considered letters of the alphabet, for a combined feature vector. Rows are targets and sum up to one, and columns are predictions.

of the screen. The bottom-right corner is used for deleting the last spelled letter from the word. Finally, when the user waves his hand towards the top-left corner, he is presented with word suggestions similar to the letters he spelled, using the Aspell dictionary library [1], effectively allowing to overcome the perceptual ambiguity in certain letters, as in Fig. 10. An example of the useage of this user interface can be seen in the video supplementing this article [13].

5. Summary & conclusions

In this article we presented a real-time method for hand-shape recognition using depth information. The proposed method shows good performance on a significant dataset including multiple users while the use of depth information provides a good robustness. The combination of appearance and depth information was shown to yield the best results. Moreover, this method is fast enough to be run real-time of a standard issue laptop. We also demonstrated the system in

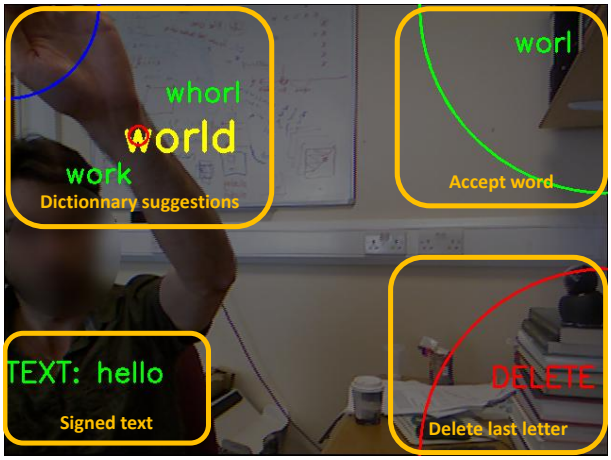


Figure 9. Illustration of the global user interface. The top-right corner validates the currently spelled word, add it to the text with a space. The bottom-right corner deletes the last letter. Finally, moving the hand towards the top-left corner displays a list of similar words gathered from a dictionary. The spelled text is written on the bottom-left.

Figure 10. Illustration of the dictionary proposal and the user selection by hand gesture. The proposals corresponding to the currently spelt word are in blue and the currently selected one (closest to the hand location) is in green, moving the hand closer to the green word will select it.

an interactive finger-spelling graphical user interface that allows the user to choose between ambiguous hypotheses and includes a dictionary for quick writing. This offers great perspectives for the practical integration of finger-spelling into computer interfaces, and the development of sign language learning tools. The current system could be improved by the inclusion of the two dynamic letters of the ASL alphabet, 'j' and 'z', and by weighting the proposed letters according to their likelihood in the word being spelled, using the dictionnary. Another interesting area for future development would be user specific adaptation by re-training the system using the user's past selections.

Acknowledgements

The research leading to these results has received funding from the European Communitys Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 231135 - DictaSign.

References

[1] Aspell. http://aspell.net. 5

[2] OpenNI. http://www.openni.org. 2, 4

[3] Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9:1545–1588, 1997. 3

[4] L. Breiman. Random forests. *Machine Learning*, pages 5–32, 2001. 3

[5] J. Daugman. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *Journal of the Optical Society of America A*, 2(7):1160–1169, 1985. 3

[6] J. Isaacs and J. Foo. Hand pose estimation for American sign language recognition. In *Proc. of Southeastern Symposium on System Theory*, pages 132–136, 2004. 2

[7] S. Liwicki and M. Everingham. Automatic recognition of fingerspelled words in British sign language. In *Proc. of CVPR*, pages 50–57, 2009. 2

[8] Microsoft. Kinect. http://www.xbox.com/kinect. 1, 2

[9] S. Mitra and T. Acharya. Gesture recognition: A survey. *IEEE Transactions on Systems Man and Cybernetics, Part C*, 37(3):311–324, 2007. 1

[10] E. Ong and R. Bowden. A boosted classifier tree for hand shape detection. In *Proc. of FGR*, pages 889–894, 2004. 2

[11] S. Ong and S. Ranganath. Automatic sign language analysis: A survey and the future beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):873–891, 2005. 1

[12] PrimeSense. NITE Middleware. http://www.primesense.com/?p=515. 2, 4

[13] N. Pugeault and R. Bowden. Spelling it out: Real–time fingerspelling recognition (video). http://www.youtube.com/watch?v=0tCGMhbTDmw, 2011. 5

[14] M. Van den Bergh and L. Van Gool. Combining RGB and ToF cameras for real-time 3D hand gesture interaction. In *Proceedings of the IEEE Workshop on Applications of Computer Vision (WACV 2011)*, 2011. 2

[15] Wikipedia. American manual alphabet. http://en.wikipedia.org/wiki/American_manual_alphabet. 2