DUODEPTH: STATIC GESTURE RECOGNITION VIA DUAL DEPTH SENSORS

Ilya Chugunov and Avideh Zakhor

University of California, Berkeley

ABSTRACT

Static gesture recognition is an effective non-verbal communication method between a user and their devices; however many modern methods are sensitive to the relative pose of the user's hands with respect to the capture device, as it can cause parts of the gesture to become occluded. We present two methodologies for gesture recognition via synchronized recording from two depth cameras to alleviate this occlusion problem. One is a more classic approach using iterative closest point registration to accurately fuse point clouds, followed by a classical PointNet architecture, and the other a dual PointNet architecture for classification without registration. On a manually collected data-set of 20,100 point clouds we show a 39.2% reduction in misclassification for the fused point cloud method, and 53.4% for the dual PointNet, as compared to a single camera.

Index Terms— Gesture recognition, point cloud, light-weight, occlusion

1. INTRODUCTION

With the ever increasing prevalence of human-computer interaction tasks, it is no surprise that gesture recognition remains a major topic of research in computer vision [1, 2, 3]. As with any vision task occlusion means a loss of data, thus degraded performance. Even without external disturbances, such as a plant in front of the camera [4], hand gestures have a tendency to self-occlude if their pose does not line up well with the geometry of the recording camera, as seen in Fig.1. For robust operation this hole in the data must be patched with another source of information.

In current literature there are three main approaches to the occlusion problem in gesture recognition: intra-frame analysis, inter-frame analysis, and sensor fusion. The first approach is well demonstrated in [4] and [5], where via processes such as integral imaging and compressive sensing the authors are able to extract more information directly from single frames, combating occlusion with algorithms that are naturally robust to it. Inter-frame analysis can be seen in papers such as [6] and [7] where multiple frames are used to provide better contextual information for gestures, alleviating the impact of temporary occlusion. The sensor fusion approach is the conceptually simplest, but least explored [8].

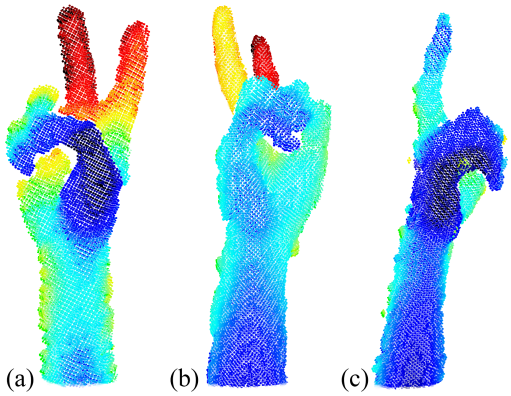

Fig. 1. Two gesture facing towards (a) left camera; (b) neither camera; and (c) right camera. Recorded from left camera.

For this paper we focus on intra-frame analysis and sensor fusion as there is limited temporal information in static gestures. Whether with multiple of the same sensor [9], or with a diverse combination [10, 11], the primary question remains the same, how to process the multiple streams of data. We propose using the PointNet architecture [12], which has been shown to perform well in problems of hand pose estimation [13, 14] as well as general object recognition [12, 15, 16]. We opt to use point clouds for the collected hand data as they are light-weight, allowing for a sparse representation of recorded data without wasting points on empty space [12, 16], and provide an intuitive way in which to fuse data from multiple camera pipelines, via coordinate system transformation and concatenation.

In this paper we outline two primary methodologies for this problem, and characterize their performance against single camera solutions:

1. Gesture recognition via single PointNet architecture on fused features derived from point clouds fused with ICP registration, referred to as FUSED.
2. Gesture recognition via dual PointNet architecture on:
   (a) fused features derived from two separate point clouds, referred to as DUAL-FEAT.
   (b) independent features derived from two separate point clouds, referred to as DUAL-CLS.

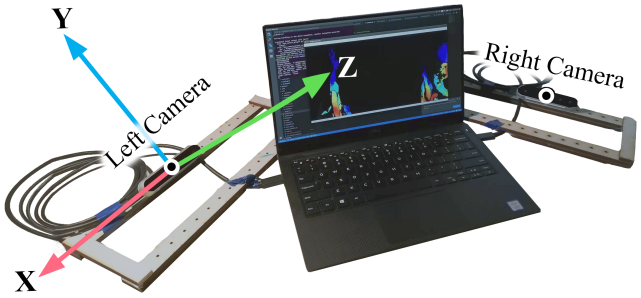

Fig. 2. Dual Intel RealSense D415 setup, XYZ axes shown.

2. DUODEPTH

2.1. Data collection

We manually recorded gestures with the setup in Fig.2, with cameras mounted perpendicular to each-other, facing inwards, and pointed 45° up from the horizontal. 1005 captures were taken of each of ten gestures, divided into 335 captures of the gesture facing towards the left camera, 335 facing the right camera, and 335 facing towards the laptop screen, i.e. neither camera. Each capture was comprised of two point clouds, one from each camera. An example of these point clouds can be seen in Fig.1.

Both depth cameras were depth limited for recording gestures, to minimize background noise, however at the start of each recording session several depth unlimited captures of the surrounding area were taken for registration purposes. The 1005 captures, two point clouds per capture, and ten gestures resulted in a total of 20,100 recorded point clouds which were then divided 80-20% into the train and test set. Unlike [17] we do not use skin color for segmentation, nor does PointNet use color, thus the gestures' RGB data is discarded.

2.2. Data fusion

In order to fuse the point clouds in each recording session for the FUSED architecture, we use an initial transform derived from the geometry of the camera setup as the seed for Open3D's color iterative closest point (ICP) function [18]. This algorithm, derived from [19], is run on the left and right depth unlimited captures and seeks to optimize:

$$E(T) = (1-\delta)E_C(T) + \delta E_G(T) \tag{1}$$

where T is the transformation matrix, δ is a weight parameter, E_C is photometric error, and E_C is geometric error as defined in Point-to-Plane ICP [20]. This allows us to find an accurate transform between the coordinate systems of the left and right depth camera for each session, which can then be used to concatenate the point clouds. If the initial geometric transform is used without ICP refinement, then small movements in camera cables and other disturbances can cause point cloud misalignments millimeters in magnitude.

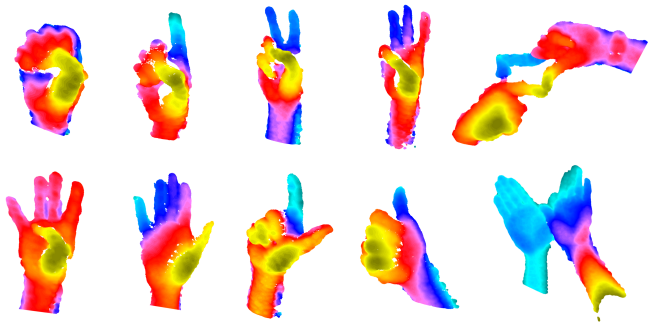

Fig. 3. The ten recorded gestures. Top left to bottom right: Zero, One, Two, Three, Frame, Four, Five, Ell, Thumb, Bird.

2.3. Point cloud processing

Although the maximum depth of the recording cameras is limited in the hardware settings ensuring the objects behind and around the user are not captured, the user's face and body are often still present in the recorded point clouds. Applying a hand isolation technique similar to [21], would not work until the samples are stripped of body and face artifacts. We found empirically that the distribution of recorded points along the Z axis, away from the camera as shown in Fig.2, provides the necessary information for cropping. Specifically there exists a high density of points on the Z axis at the face of the gestures, such as the palm for the Five gesture. There is also a high density of points at the location of the body/face, if it is present in the point cloud, leading to a multimodal distribution of points along the Z axis. Thus, applying:

$$detect\_peaks(data_Z) \rightarrow \begin{cases} \#peaks = 1, & \text{pass} \\ \#peaks = 2+, & \text{isolate arm} \end{cases} \tag{2}$$

would result in point clouds of only the user's arm, which could then undergo fine processing [21] to isolate the hand. Here $data_Z$ can be any measure of point density along the Z axis, such as a histogram on the Z data of the point cloud.

Similar to [12, 22, 23], we found that data augmentation proved to be beneficial for classification accuracy, helping combat over-fitting in high epoch training. Points were randomly jittered as in [12], and random translations were performed on the point cloud as a whole. Rotational augmentation was not used, as even at small magnitudes of $< 5°$ it invariably lowered test accuracy. Thus the augmentation of a point cloud with points at (X^N, Y^N, Z^N) takes the form:

$$\widetilde{P}^N = (X^N, Y^N, Z^N) + (J_x^N, J_y^N, J_z^N) + (T_x, T_y, T_z)^N$$
$$T_x, T_y, T_z \sim \mathcal{N}(0, \alpha) \qquad J_x^N, J_y^N, J_z^N \sim \mathcal{N}_N(0, \beta) \tag{3}$$

where $\widetilde{P} = (\tilde{X}^N, \tilde{Y}^N, \tilde{Z}^N)$ is the new point cloud of size N, J is the jitter size, T is the translation size, and α and β are empirically determined as 0.002 and 0.01 in Section 3.2.

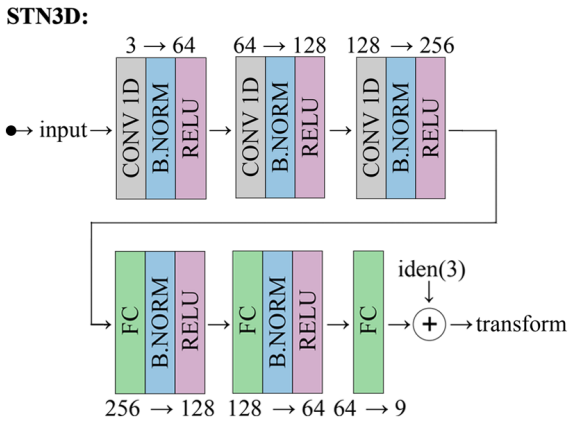

Fig. 4. 3D spatial transformer network (**STN3D**).

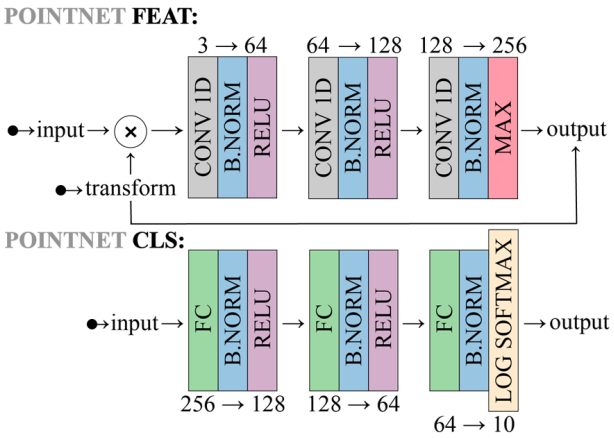

Fig. 5. Feature extractor (**FEAT**) and classifier (**CLS**).

2.4. Network architecture

The network architectures in this paper are each constructed from three base components: the 3D spatial transformer network, feature extractor, and classifier. The transformer network, shown in Fig.4, is a modified version of STN3D [24], which itself is a 3D adaptation of the methods of [25]. It outputs per-sample transformations to be used in the feature extraction layers, and serves to produce a final model invariant to rotations, translations, and changes of scale, acting as the T-net [12]. The feature extractor (FEAT) and classifier layers (CLS), shown in Fig.5, are direct adaptations of the PointNet architecture [12] [26], with significantly reduced layer sizes.

*2.4.1. Fused features, fused point clouds (**FUSED**)*

For the FUSED architecture the input is a single fused point cloud containing information from both the left and right cameras, as described in Section 2.3. We thus use a single PointNet pipeline:

$$\begin{aligned} &\quad\quad\quad\quad input \searrow \\ input \rightarrow STN3D \rightarrow FEAT \rightarrow CLS \rightarrow output \end{aligned} \tag{4}$$

A 3D spatial transform is calculated for the fused point cloud as a whole, applied, and the output is then passed into the feature extraction layers.

*2.4.2. Fused features, separate point clouds (**DUAL-FEAT**)*

In the DUAL-FEAT architecture the input is two separate point clouds which are to be fused at the feature level:

$$\begin{aligned} &\quad\quad\quad\quad input1 \searrow \\ input1 \rightarrow STN3D \rightarrow FEAT \rightarrow CLS \rightarrow output \\ input2 \rightarrow STN3D \nearrow \\ &\quad\quad\quad\quad input2 \nearrow \end{aligned} \tag{5}$$

3D spatial transforms are calculated independently for each input point cloud, applied, and the two inputs are fused before being passed into the feature extraction layers.

*2.4.3. Separate features, separate point clouds (**DUAL-CLS**)*

In this third architecture, DUAL-CLS, the input is two separate point clouds whose features are to to be calculated independently and fused in the classifier:

$$\begin{aligned} &\quad\quad\quad\quad input1 \searrow \\ input1 \rightarrow STN3D \rightarrow FEAT \rightarrow CLS \rightarrow output \\ input2 \rightarrow STN3D \rightarrow FEAT \nearrow \\ &\quad\quad\quad\quad input2 \nearrow \end{aligned} \tag{6}$$

This is identical to (5) save for the fact that the computed features are concatenated rather than the inputs fused.

3. EXPERIMENTS

3.1. Implementation and runtime

Network code is adapted from [26], a re-implementation of PointNet in PyTorch, and was executed on a machine with an Intel Core i7-6850K CPU @ 3.60GHz and a single GeForce GTX 1080Ti. Code and data-set are available at [27].

All pipelines required an average of 2.9ms to crop, 3.0ms to down-sample, and 1.2ms to classify a single input. A total of 7.1ms per input, significantly lower than the 20ms+ required by works such as [28, 29]. Cropping and down-sampling times can be further improved to under a millsecond each by limiting point cloud acquisition size in the hardware of the cameras, however the effect of this on overall classification accuracy remains untested.

3.2. Ablation testing

In [12], the authors found that performance improved as the number of points in the input point clouds increased, however noted little to no improvement from using over one thousand points. We ran a similar series of tests, displayed in Fig.6, and observed improvements in overall classification accuracy up to a size of 320, which was chosen as N, the final point cloud

Pipeline Variant	Overall Mean	Zero Mean	One Mean	Two Mean	Three Mean	Four Mean	Five Mean	Thumb Mean	Ell Mean	Frame Mean	Bird Mean
Left Only	0.878	0.873	0.840	0.939	0.905	0.850	0.724	0.934	0.828	0.968	0.921
Right Only	0.885	0.886	0.854	0.840	0.880	0.867	0.764	0.954	0.889	0.976	0.937
FUSED	0.928	0.968	0.909	0.918	0.893	0.942	**0.801**	0.971	0.933	0.990	0.954
DUAL-FEAT	**0.945**	**0.972**	**0.931**	**0.975**	0.941	**0.979**	0.771	0.970	**0.953**	0.995	**0.963**
DUAL-CLS	0.938	0.938	0.919	0.971	**0.967**	0.932	0.791	**0.972**	0.932	**0.997**	0.962

Table 2. Accuracy results for various inputs and architectures for: 100 trials, 100 epochs, 8040 train, 2010 test

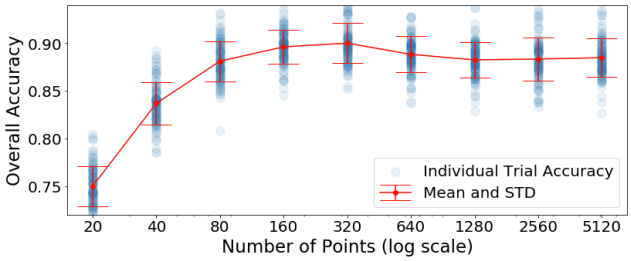

Fig. 6. Point cloud size versus mean accuracy for: 100 trials, 32 epochs, 8040 train, 2010 test, and FUSED architecture.

size for training and testing. Augmentation magnitudes α and β, used in Equation (3), were chosen via a similar process.

As shown in Table 1 a number of augmentations of the original PointNet architecture were tested, including: (D) dropout in the second fully connected layer in the classifier shown in Fig.5, (R) reduction of convolution and fully connected layer sizes, and (SELU) self normalizing functions in place of batch normalization and RELU [30]. Due to our use of smaller point clouds than in [12] for training and testing, the largest improvement in accuracy was achieved from reduction of the layer sizes, or equivalently the number of network parameters.

3.3. Results

Table 2 showcases the final test results of the three dual camera pipelines as well as two single camera pipelines, which were both trained on the standard PointNet architecture in Equation (4). The left and right cameras performed similarly overall, however presented noticeable classification disparities in more asymmetric gestures such as Two, Five, and Ell. The FUSED pipeline produced a 39.2% reduction in misclassification when compared to averaged single camera performance. DUAL-FEAT and DUAL-CLS also showed 53.4% and 47.7% reductions in misclassification respectively. DUAL-FEAT achieves top performance in the majority of gestures, as it is able to more effectively classify captures where both the left and right camera receive occluded data and form incorrect independent features for classification by DUAL-CLS.

When compared to other modern methods for static gesture recognition with depth data, as shown in Table 3, our

Augmentation:	NONE	D	R	R+D	SELU
Mean Accuracy:	0.899	0.903	**0.907**	0.885	0.858

Table 1. Network augmentation ablation results for: 300 trials, 64 epochs, 8040 train, 2010 test, and FUSED architecture.

Method:	[28]	DUAL-FEAT	[31]	[10]	[7]
Mean Accuracy:	**0.999**	0.945	0.942	0.913	0.844
Train Size:	28K	20K	26.7K	1.4K	26.7K
Data-set Available:	✗	✓	✓	✓	✓

Table 3. Comparison of mean accuracy for 10 static gesture classification.

single camera test results are noncompetitive, as the data-set in this paper is purposely created to be challenging for a single view pipeline. The DUAL-FEAT mean accuracy is however quite similar to [31], which uses 4 point clouds collected temporally from a single depth camera for classification. The point cloud samples in [31] are created with the test subject always facing towards the camera, favourably to its geometry, and as such the methods of [31] are not likely to perform as well on a more challenging self-occluded data-set such as the one presented in this paper. Additionally [31] uses 2048 points per point cloud, a total of 12.8 times more points per input than DuoDepth, requiring a greater number of network parameters to train and significantly more computation for classification.

Max Rotation (ζ):	1°	5°	10°	15°	20°
FUSED Mean:	0.854	0.555	0.499	0.334	0.313
DUAL-FEAT Mean:	0.838	0.541	0.415	0.347	0.340
DUAL-CLS:	**0.907**	**0.767**	**0.836**	**0.838**	**0.849**

Table 4. Rotational augmentation results for various architectures for: 100 trials, 32 epochs, 8040 train, 2010 test.

As mentioned in Section 2.3, adding rotational augmentation invariably lowered test accuracy. Purposely applying random rotations of magnitude $\mathcal{N}(0, \zeta)$ to left and right input independently it was observed that Dual-CLS had a significantly higher resilience to rotational noise than the other two camera pipelines, shown in Table 4. This presents interesting potential applications for DUAL-CLS such as multi-robot gesture recognition. If there are no good estimates of relative poses between robots' on-board cameras, large random rotations are naturally introduced over time in their operation.

4. REFERENCES

[1] Rafiqul Zaman Khan and Noor Ibraheem, "Hand gesture recognition: A literature review," *International Journal of Artificial Intelligence Applications (IJAIA)*, vol. 3, pp. 161–174, 08 2012.

[2] Hong Cheng, Lu Yang, and Zicheng Liu, "Survey on 3d hand gesture recognition.," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 26, no. 9, pp. 1659–1673, 2016.

[3] M. Asadi-Aghbolaghi, A. Claps, M. Bellantonio, H. J. Escalante, V. Ponce-Lpez, X. Bar, I. Guyon, S. Kasaei, and S. Escalera, "A survey on deep learning based approaches for action and gesture recognition in image sequences," in *2017 12th IEEE International Conference on Automatic Face Gesture Recognition (FG 2017)*, May 2017, pp. 476–483.

[4] V. J. Traver, P. Latorre-Carmona, E. Salvador-Balaguer, F. Pla, and B. Javidi, "Three-dimensional integral imaging for gesture recognition under occlusions," *IEEE Signal Processing Letters*, vol. 24, no. 2, pp. 171–175, Feb 2017.

[5] H Zhuang, M Yang, Z Cui, and Q Zheng, "A method for static hand gesture recognition based on non-negative matrix factorization and compressive sensing," vol. 44, pp. 52–59, 01 2017.

[6] M. Madadi, S. Escalera, A. Carruesco, C. Andujar, X. Bar, and J. Gonzlez, "Occlusion aware hand pose recovery from sequences of depth images," in *2017 12th IEEE International Conference on Automatic Face Gesture Recognition (FG 2017)*, May 2017, pp. 230–237.

[7] Joshua Owoyemi and Koichi Hashimoto, "Spatiotemporal learning of dynamic gestures from 3d point cloud data," *CoRR*, vol. abs/1804.08859, 2018.

[8] Raffaele Gravina, Parastoo Alinia, Hassan Ghasemzadeh, and Giancarlo Fortino, "Multi-sensor fusion in body sensor networks: State-of-the-art and research challenges," *Information Fusion*, vol. 35, pp. 68 – 80, 2017.

[9] Samarjit Das Inkyu Moon Tabassum Nasrin, Faliu Yi, "Partially occluded object reconstruction using multiple kinect sensors," 2014.

[10] G. Marin, F. Dominio, and P. Zanuttigh, "Hand gesture recognition with leap motion and kinect devices," in *2014 IEEE International Conference on Image Processing (ICIP)*, Oct 2014, pp. 1565–1569.

[11] K. Liu, C. Chen, R. Jafari, and N. Kehtarnavaz, "Fusion of inertial and depth sensor data for robust hand gesture recognition," *IEEE Sensors Journal*, vol. 14, no. 6, pp. 1898–1903, June 2014.

[12] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *CoRR*, vol. abs/1612.00593, 2016.

[13] Liuhao Ge, Yujun Cai, Junwu Weng, and Junsong Yuan, "Hand pointnet: 3d hand pose estimation using point sets," in *CVPR*, 2018.

[14] Liuhao Ge, Zhou Ren, and Junsong Yuan, "Point-to-point regression pointnet for 3d hand pose estimation," in *ECCV*, 2018.

[15] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *CoRR*, vol. abs/1706.02413, 2017.

[16] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas, "Frustum pointnets for 3d object detection from rgb-d data," *arXiv preprint arXiv:1711.08488*, 2017.

[17] Jiaming Li, Yulan Guo, Yanxin Ma, Min Lu, and Jun Zhang, "Integrating color and depth cues for static hand gesture recognition," 11 2017, pp. 295–306.

[18] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun, "Open3d: A modern library for 3d data processing," *CoRR*, vol. abs/1801.09847, 2018.

[19] Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun, "Colored point cloud registration revisited," *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 143–152, 2017.

[20] Yang Chen and Grard Medioni, "Object modelling by registration of multiple range images," *Image and Vision Computing*, vol. 10, no. 3, pp. 145 – 155, 1992, Range Image Understanding.

[21] Bingyuan Xu, Zhiheng Zhou, Xi Chen, Yi Yang, and Zhiwei Yang, "Arm removal for static hand gesture recognition," *Journal of Intelligent Fuzzy Systems*, pp. 1–12, 11 2018.

[22] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2015, pp. 922–928.

[23] Lyne P. Tchapmi, Chris Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese, "Segcloud: Semantic segmentation of 3d point clouds," 10 2017.

[24] Shubham Tulsiani, "stn3d," https://github.com/shubhtuls/stn3d, 2018.

[25] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu, "Spatial transformer networks," *CoRR*, vol. abs/1506.02025, 2015.

[26] Fei Xia, "pointnet.pytorch," https://github.com/fxia22/pointnet.pytorch, 2018.

[27] Ilya Chugunov, "Duodepth," https://github.com/Ilya-Muromets/DuoDepth, 2018.

[28] Chaoyu Liang, Yonghong Song, and Yuanlin Zhang, "Hand gesture recognition using view projection from point cloud," 09 2016, pp. 4413–4417.

[29] Eriglen Gani and Alda Kika, "Albanian sign language (albsl) number recognition from both hands gestures acquired by kinect sensors," *International Journal of Advanced Computer Science and Applications*, vol. 7, 08 2016.

[30] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter, "Self-normalizing neural networks," *CoRR*, vol. abs/1706.02515, 2017.

[31] Cherdsak Kingkan, Joshua Owoyemi, and Koichi Hashimoto, "Point attention network for gesture recognition using point cloud data," in *BMVC*, 2018.