**Topic Identification of Filipino and English News using Bidirectional Long-Short Term Memory with Attention Mechanisms**

ABSTRACT

In this paper, a new approach to American Sign Language is created. By using a web camera or a mobile camera with 91 gestures, each composed of fingerspelling and dynamic gestures. The research aims to surpass the environmental limitations of webcam-related ASL recognition, which only shows the hand of the given signer. This study focuses on solutions that utilizes normal cameras.

The past related studies that used normal cameras such as web cameras, are often limited by constraints on the environment perceived by the camera. Some solutions require the presence of a solid black background while others require the hand to be well lit. there are also solutions that use accessories that serve as tracking markers to aid the camera. The most common limitation among these solutions is that the hands are the only parts of the body perceived by the camera.

By using Pose-Estimation with Part Affinity Fields to overcome the environmental limitations, and Hidden Markov Models for the recognition of a gesture, the research has obtained an accuracy of 64% for fingerspelling gestures, 43% for dynamic gestures, and 36% when the previous two cases are combined.

Albeit falling behind in terms of accuracy compared to researches that used Microsoft Kinect or any similar camera devices augmented with depth sensors, this research shows a possibility of using a robust feature extraction library with a time-series classification algorithm like the Hidden Markov Model to be able to replicate the performance of gesture systems that utilize Kinect.

KEYWORDS

American Sign Language (ASL) Recognition, Pose Estimation, Hidden Markov Models, Gesture Recognition

1 Introduction

There have been students in the fields of computer vision and pattern recognition that provide solutions that aims to solve American Sign Language Recognition. Some solutions utilize devices such as data gloves which are special gloves that are made to track a person’s hands, special 3D cameras that uses depth and image video feeds to track motion easily, and normal cameras which are found on today’s common gadgets such as phones, laptops, and many more.

This study focuses on solutions that utilizes normal cameras. The past related studies that used normal cameras such as web cameras, are often limited by constraints on the environment perceived by the camera. Some solutions require the presence of a solid black background while others require the hand to be well lit. there are also solutions that use accessories that serve as tracking markers to aid the camera. The most common limitation among these solutions is that the hands are the only parts of the body perceived by the camera.

The study aims to overcome these environmental limitations by implementing a system that makes use of *Pose Estimation* with Part Affinity Fieldsas the preprocessing algorithm and the *Hidden Markov Models* for the recognition of the ASL gesture.

1.1 Statement of the Problem

The study of Topic Identification using Attention Mechanisms proposes a new system for machine interpretability of Long-Short Term Memory Networks. This is done by applying the Attention Mechanisms proposed by the research of [] to Topic Identification. The broad problem of applying attention mechanisms to this problem branches out to the following more specific subproblems:

1. Will the new proposed system be accurately able to classify Filipino and English news using attention mechanisms?

2. What features/words are important to classify the given input?

2 Background of Study

For the input, the most common device used is a standard camera, either through an image or a video. However, there are researchers that made use of the Microsoft Kinect [1,3,5,6,7,12,14]. The Kinect is equipped with a depth sensor along with a standard camera. By using the data from the sensor and the camera, the process for hand recognition can be simplified.

For the process, the most common method used is the Hidden Markov Model [6,7,13] A new HMM has been proposed in which is called Fast HMM by [14]. Neural Networks by 8], as well as Artificial Neural Networks (ANN)[9], Dynamic Time Warping (DTW) [3, and other algorithms for recognition are used in other researches.

In [2] developed Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. Pose Estimation is the localization of the anatomical parts of a person. [4] developed a 3D Model Pose Estimation of a hand using a 3D neural network. In the same year, [11], together with [2] developed a hand pose estimation algorithm using multiple cameras.

Using the researches of [2] and [11] as the basis, a library called OpenPose was developed, which can detect hand, body, and face at the same time. However, 3D models require higher computing power than 2D Models, which is why 2D Pose Estimation was used to obtain the input of this study.

In this research, the OpenPose library is used to recognize the limbs of the signer. The output of the methods provided by the library will be used as the input for the HMM, the classifier for the gestures. According to one of the researches that served as the basis for OpenPose, the algorithm can yield accurate results regardless of location, orientation, and environment background [2].

3 Methodology

The system was written in the *Python* programming language. The researchers made use of its 3.6 release (*Python 3.6.x)*. Several *Python* libraries are required to make the system work, such as *Chainer, CuPy, Pandas, and HMMlearn.* These libraries are the main modules used for the system. There are several other modules that are needed, but this section will only highlight the core ones.

*Nvidia CUDA* was utilized to dramatically speed up processing time by leveraging the capabilities of the graphics card of the laptop. For the system to be able to utilize CUDA functionalities, the *Nvidia CUDA Toolkit 9.2* is also required. It should be noted that if one wishes to replicate the study using another graphics card, it should be able to support *Nvidia CUDA.* Installing the *CUDA toolkit* requires the latest *Microsoft Visual Studio* to be installed as well. It should be greatly noted that *Visual Studio* must have the *Visual Studio 2015 C++ (version 14.0)* module installed.

For the implementation of the system, a powerful laptop was used. The laptop that was used was equipped with a 7th-generation *Intel i5* processor, 8 gigabytes of RAM, and a *Nvidia GeForce GTX1050* graphics card. The laptop also had a built-in webcam.

Powerful hardware is recommended because the *OpenPose* library demands significant amounts of computing power in order to work in real-time. When the videos were pre-processed into key points, it took almost a day with the laptop that the researchers used.

To obtain the videos that were used for the dataset, the researchers asked for the help of several people to serve as actors that perform the gestures. At least 10 actors were involved in the videos, including some of the researchers.

The total number of gestures is 91 ASL gestures, with each actor performing all 91 gestures. However, there are other videos that are inconsistent with other videos, which is considered as wrongly recorded, thus removed. Overall, there are a total of 900+ videos for the dataset that is used in this research.

3.1 Input

For the input, the most common device used is a standard camera, a video from a web camera. A video with a 720p quality, with the signer perpendicular to the webcam and only the upper body must be seen in the video.

However, there are differences on the source where the videos will come from. For the training and testing phase, the videos will be coming from the dataset. For the using phase, an internal or external webcam will be the source of the videos.

3.2 Pre-Processing

For the Pre-Processing Module, an implementation of [2] using *Chainer*, a Python library is used to extract the body key points and hand key points from the signer. The graphical representation of these keypoints are shown on Figure 1 and 2. There are a total of 56 keypoints, a total of 112 values because of the keypoints having x and y values, Principal Component Analysis is used to convert each x and y keypoint to only one value, thus having a total of 56 keypoints that serves as the features of each frame.

The program goes through each video in a frame-by-frame fashion, capturing only frames after an interval of frames has passed. This interval is calculated by the program by dividing the length of the video by 12; the pre-processing program is only set to use the feature extraction function of the *OpenPose* library for only 12 frames of a video. Per frame that was read by the function of the *OpenPose* library.

Figure 1: Representation of the keypoints for *‘hello’*

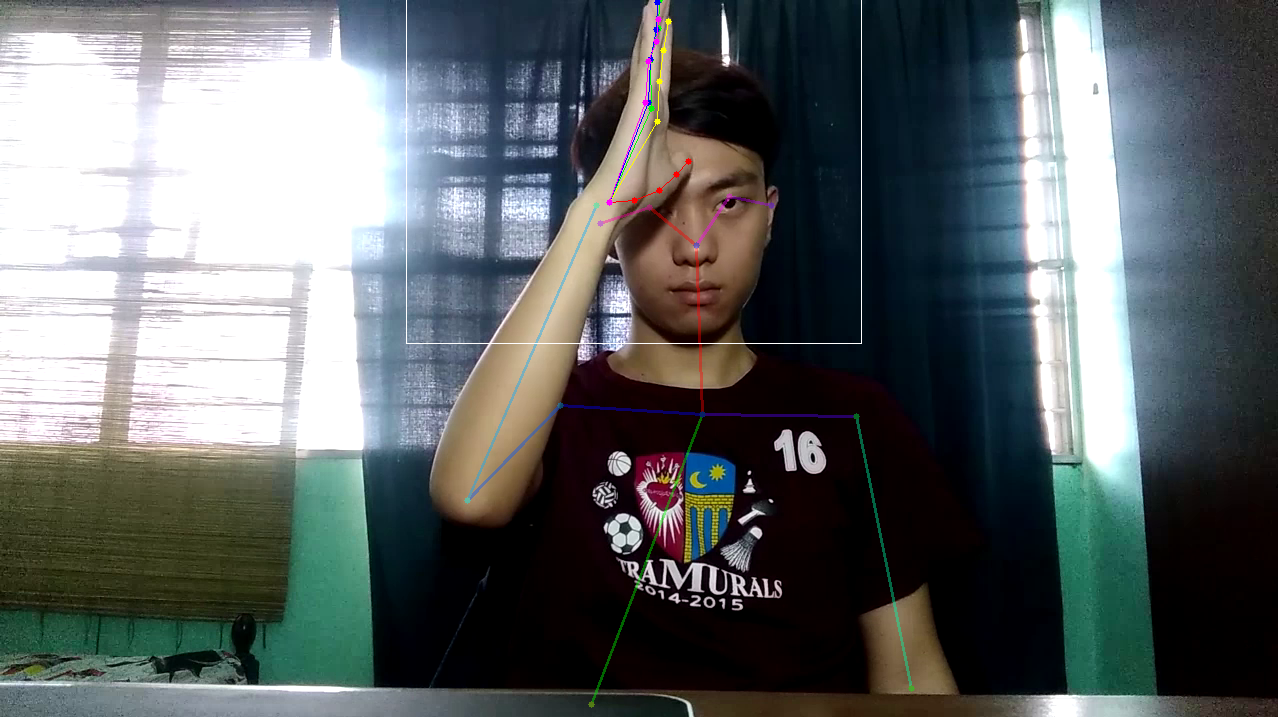
 

Figure 2: Representation of the keypoints for *‘father’*

3.1 Process

After extracting the key points, the system architecture mentioned by [10] is implemented for the Hidden Markov Model using the *HMMlearn* library in Python, which is shown in Figure 3. The implementation from [10] is for speech recognition, but it can also be used in other related problems such as Gesture Recognition, which falls in this research.

A close up of a logo

Description automatically generated

Figure 3: HMM Implementation

A close up of a black background

Description automatically generated

Figure 4: System Architecture Diagram

As seen in Figure 4, the system is comprised of three phases: training, testing, and using. The goal of the first phase, which is the training phase, is to create multiple *Hidden Markov Model* (HMM) instances, and to make each instance learn the probability values for their respective gestures in order to make predictions. To teach each instance their probability values, the *Baum-Welch algorithm* is implemented.

The dataset is a csv file composed of all the extracted key points from all the 900+ videos available in the dataset

The training phase is directly connected to the testing phase. After the HMM instances have been trained by the *Baum-Welch algorithm*, the said instances can now use the *Viterbi algorithm* to classify what gesture the input is most likely to be. The testing phase is meant to determine the accuracy of the system. The accuracy can be determined by how many times the system predicted the gesture from the videos correctly. Both the training and the testing phase makes use of pre-stored videos from the dataset as input. It should be noted that all videos that are used in this system are pre-processed by the *OpenPose library* before being fed into HMM-related algorithms.

Once the training phase has yielded satisfactory results, the system can now be exposed to real-world cases; the system is now under the using phase. Like the training phase, the using phase makes use of the *Viterbi algorithm* to classify gestures. The difference of this phase from the previous two is that it does not make use of pre-stored videos from the dataset, instead, it accepts videos captured by an internal or external webcam.

After all the HMMs are trained, the input from each data are fed to each of the trained Hidden Markov Models for testing. By using the score() method for the input to each of the hidden Markov model, the likelihood of a given sequence is obtained, after getting the maximum likelihood, the *Bayesian Information Criterion* is applied to each of the value from the score() method. The model who has the highest value is selected for prediction.

The data is split to 50% for training data, and 50% for test data, since there is a minimum of 10 actors for each gesture, there are 5 actors for each gesture for training data, and 5 speakers for test data, the 10 actors per gesture are shuffled before splitting into training data and test data.

60%-40%, 70%-30%. 80% -20% train-test split is also experimented. Cross validation is also tried for the given data sets at k-fold = 2, k-fold = 3, k-fold = 4.

Different test cases are tried by segmenting the given gestures:

1. Combination of finger spelling gestures and dynamic gestures (93 gestures in total)
2. Dynamic gestures only (56 gestures in total)
3. Finger spelling gestures only (33 gestures in total)
4. Splitting dynamic gestures into two sets (28 gestures each set)

Since HMMs has a designated number of states, the training and testing phase is repeated at hidden states = 1 up to hidden states = 25 for each test cases. The state with the highest accuracy is selected as the final model for ASL Recognition.

4 Results and Discussions

Figure 5 gives an overview of all 5 test cases. It shows the accuracy of each hidden state configuration for each test case.

It can be observed from the trend that the fingerspelling-only test case (dark blue) is positioned in the higher part of the graph. It can also be observed that the red line belonging to the fingerspelling + dynamic test case is positioned on the lower part of the graph. Of all the test cases, the fingerspelling-only set yielded the highest accuracy of 64%, whilst the fingerspelling + dynamic test case yielded a low accuracy of 36.74%.

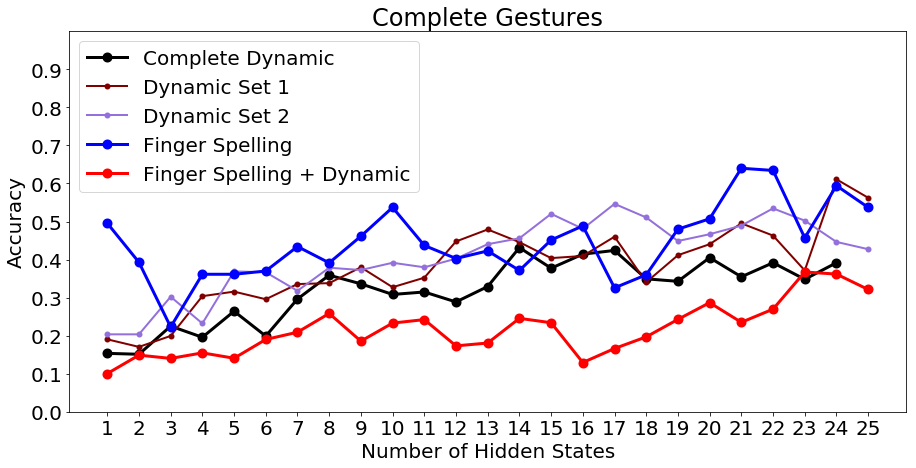


Figure 5: Comparison Between Different Test Cases and Their Accuracies at Different Hidden States

From the different train-test splits and k-fold cross validations, it shows that the 50%-50% train-test split is the best case for the data. Although the test case for the dynamic set 2 is the best for 2-Fold cross validation and the test case for the complete dynamic gestures is also best for 2-Fold cross validation, with only 3% and .80% difference from the 50%-50% train-test split, respectively, the research needs to stick to one train-test case to be presented, and through simple majority, the 50%-50% train-test split is chosen as the best case.

It can be noticed on Figure 5, the greater number of states, the greater the accuracy will be, however some test cases like the finger spelling test case, fluctuates at different number of states. From states at 1-25, most test cases get their best accuracy at 23. 25 is the limit since some test cases throw an error at state=24 or 25. This is due to the data fed to the HMMs. Since the data is limited to 12 frames per gesture. If there are more frames, there can be higher number of states. Since each frame corresponds to a state in an HMM.

**Table(s) 1-5: Summary Between Different Test Cases on Different Test Methods**

**Dynamic Gestures**

|  |  |  |
| --- | --- | --- |
| **Test Method** | **Accuracy** | **Precision** |
| 50%-50% Train-Test Split | 43.15% | 49.47% |
| 60%-40% Train-Test Split | 32.14% | 39.70% |
| 70%-30% Train-Test Split | 20.24% | 23.08% |
| 80%-20% Train-Test Split | 17.01% | 38.75% |
| **2-Fold Cross Validation** | **43.95%** | **73.79%** |
| 3-Fold Cross Validation | 17.01% | 38.75% |
| 4-Fold Cross Validation | 14.75% | 37.93% |

Fingerspelling Gestures

|  |  |  |
| --- | --- | --- |
| **Test Method** | **Accuracy** | **Precision** |
| **50%-50% Train-Test Split** | **64.00%** | **63.01%** |
| 60%-40% Train-Test Split | 50.00% | 54.86% |
| 70%-30% Train-Test Split | 40.00% | 44.92% |
| 80%-20% Train-Test Split | 31.43% | 36.93% |
| 2-Fold Cross Validation | 55.84% | 74.02% |
| 3-Fold Cross Validation | 39.96% | 78.96% |
| 4-Fold Cross Validation | 27.50% | 62.39% |

Complete Gestures

|  |  |  |
| --- | --- | --- |
| **Test Method** | **Accuracy** | **Precision** |
| **50%-50% Train-Test Split** | **36.74%** | **40.35%** |
| 60%-40% Train-Test Split | 25.11% | 29.23% |
| 70%-30% Train-Test Split | 15.93% | 16.33% |
| 80%-20% Train-Test Split | 9.16% | 8.63% |
| 2-Fold Cross Validation | 33.51% | 58.30% |
| 3-Fold Cross Validation | 16.97% | 35.06% |
| 4-Fold Cross Validation | 10.41% | 28.47% |

**Dynamic Set 1**

|  |  |  |
| --- | --- | --- |
| **Test Method** | **Accuracy** | **Precision** |
| **50%-50% Train-Test Split** | **61.11%** | **65.27%** |
| 60%-40% Train-Test Split | 37.32% | 40.50% |
| 70%-30% Train-Test Split | 27.38% | 31.55% |
| 80%-20% Train-Test Split | 19.64% | 12.39% |
| 2-Fold Cross Validation | 46.27% | 67.06% |
| 3-Fold Cross Validation | 27.89% | 55.71% |
| 4-Fold Cross Validation | 10.41% | 28.47% |

**Dynamic Set 2**

|  |  |  |
| --- | --- | --- |
| **Test Method** | **Accuracy** | **Precision** |
| 50%-50% Train-Test Split | 54.64% | 60.52% |
| 60%-40% Train-Test Split | 50.71% | 52.55% |
| 70%-30% Train-Test Split | 39.58% | 36.90% |
| 80%-20% Train-Test Split | 28.57% | 18.44% |
| **2-Fold Cross Validation** | **57.58%** | **79.19%** |
| 3-Fold Cross Validation | 36.31% | 74.55% |
| 4-Fold Cross Validation | 27.26% | 65.67% |

Summarizing the results from the different test cases, the highest accuracy came from the fingerspelling gestures with an accuracy of 64%, this is since fingerspelling gestures are simpler than the dynamic gestures. For the complete dynamic gestures, a 43% accuracy is achieved with 56 total gestures, the researchers tried to lessen the number of gestures to be recognized by splitting the dynamic gestures into two with an accuracy of 61% and 54% respectively. By combining the total fingerspelling gestures and dynamic gestures, the accuracy lowered down to 36%. This proves that the lesser the number of gestures to be recognized, the more accurate the model will be. Additionally, the number of frames taken can be the reason why there is a low accuracy for the different test cases.

A reason why a low accuracy is obtained from different test cases can be traced back to the data, since only 12 frames per input is obtained, there is a heavy data loss after the pre-processing algorithm. However, even if there is a heavy data loss, this research still achieved the highest accuracy of 64% in one of the test cases.

A possible factor that contributed to the research yielding low accuracy is that the data is not normalized. The keypoints that the *OpenPose* library produces are x and y axes with respect to the video resolution. However, since not all actors are perfectly centered, it can cause the HMMs to produce different probability values. For example, if the actor is positioned more on the right, the x value of the coordinates will increase. If the actor is positioned more on the left, the x values will decrease. Such difference in positioning can cause noise in the data, especially if the position of the actors is inconsistent, thus reducing the accuracy since it can throw of the probability computations of the HMMs.

5 Summary and Conclusions

American Sign Language Recognition is a challenge in computer vision, since it can help the deaf community communicate between non-deaf people using artificial intelligence.

The study creates a program that can potentially bridge the barrier between normal people and those with hearing or speaking disorders that rely on ASL as their sole form of communication. Compared to some of the previously created systems, it does not make use of Microsoft Kinect, any form of special gloves, or other bulky equipment.

The research was also able to recognize almost accurately the 56 dynamic gestures and 35 fingerspelling hand signs from the American Sign Language with different test cases. For the fingerspelling hand signs, the research achieved an accuracy of 64% while the 56 dynamic gestures achieved an accuracy of 43.15%.

This research as proved that there is a possibility of creating an ASL recognition system where the input videos of signers have unrestricted environment -- the signer has no restrictions on what background will be present in the video and the upper half body of the signer can be visible. The research achieved this environment improvement by using only a standard optical webcam.

6 Recommendations

### Normalization

To improve the accuracy yielded by the system, the researchers recommend normalizing the data. By normalizing the data, the accuracy can further increase because the position of the actor in the video can cause little to no impact to the computation of the HMMs since it will cause less noise in the data.

### Increasing the Number of Frames

One of the things that the researchers strongly recommend is increasing the number of frames extracted from the video. From jumping to 5 *frames per second* from 12 *frames per video,* information loss can be mitigated greatly. 12 frames per second causes information loss, and even further information loss due to the pre-processing stage possibly capturing blurry frames.

### Increasing the Number of Videos in the Dataset

Lastly, the researchers recommend that the number of actors be increased. More video to work with means more data. A higher amount of data can mean increased accuracy when performing tests with test-train splits or k-fold cross validations.

ACKNOWLEDGMENTS

The researchers would like to thank our Almighty God, for His blessing and guidance throughout this research. The researchers would also like to thank their respective families, friends, and relatives who gave their whole support throughout the creation of this research.

Thank you also to Mr. Jessie James Suarez, Prof. Jordan Deja, who made themselves available when the researchers need advice and guidance with the codes, and implementations of the algorithms of this research.

The researchers cannot complete this research without the persons who served as our actors.

To our beloved University of Santo Tomas, continue showering future researchers with your quality of education.

*Always stand on the shoulder of giants*

REFERENCES

[1] S. Rajaganapathy, B. Aravind, B. Keerthana, and M. Sivagami. 2015. Conversation of Sign Language to Speech with Human Gestures. Procedia Computer Science 50 (2015), 10–15. DOI: http://dx.doi.org/10.1016/j.procs.2015.04.004

[2] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. 2017. Realtime Multi-person 2D Pose Estimation Using Part Affinity Fields. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017). DOI: http://dx.doi.org/10.1109/cvpr.2017.143

[3] Esther F. Casabar, Hermie N. Prim, Maria B. Santos, Charmaine D. Tan. 2014. American sign language translation through skeletal and head point tracking using dynamic time warping. Undergraduate Thesis. University of Santo Tomas (UST), Espana, Manila.

[4] Xiaoming Deng, Shuo Yang, Yinda Zhang, Ping Tan, Liang Chang, and Hongan Wang. 2007. Hand3D: Hand Pose Estimation using 3D Neural Network. arXiv:1704.02224. Retrieved from https://arxiv.org/abs/1704.02224

[5] Jordan Aiko Deja, Eric Espiridion, Ryan Carlo Magimot, and Neil Anthony Rosales. 2015. A Framework for a Customizable Sign Language to Text Transcriber Prototype. (August 2015). Retrieved March 17, 2018 from https://www.researchgate.net/publication/281031359\_A\_Framework\_for\_a\_Customizable\_Sign\_Language\_to\_Text\_Transcriber\_Prototype

[6] Debi Prosad Dogra, Himaanshu Gauba, Pradeep Kumar, and Partha Pratim Roy. 2017. Coupled HMM-based multisensor data fusion for sign language recognition. Pattern Recognition Letters, 86 (January 2017), 1-8. DOI: https://doi.org/10.1016/j.patrec.2016.12.004

[7] Pradeep Kumar, Partha Pratim Roy, and Debi Prosad Dogra. 2018. Independent Bayesian classifier combination based sign language recognition using facial expression. Information Sciences 428, C (February 2018), 30-48. DOI: https://doi.org/10.1016/j.ins.2017.10.046

[8] Olivier Gibaru, Pedro Neto, and Miguel Simão. 2017. Using data dimensionality reduction for recognition of incomplete dynamic gestures. Pattern Recognition Letters 99, (November 2017), 32-38. DOI: https://doi.org/10.1016/j.patrec.2017.01.003

[9] Mohiminul Islam, Sarah Siddiqua, Jawata Afnan. 2017. Real time Hand Gesture Recognition using different algorithms based on American Sign Language). 2017 IEEE International Conference on Imaging, Vision and Pattern Recognition (icIVPR), 2,4. DOI:10.1109/ICIVPR.2017.7890854

[10] Lawrence Rabiner. 1989. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE, 11, 2 (February 1989), 257-286

[11] Hanbyul Joo, Iain Matthews, Yaser Sheikh, and Tomas Simon. 2017. Hand Keypoint Detection in Single Images Using Multiview Bootstrapping. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). arXiv:1704.07809. Retrieved from: https://arxiv.org/abs/1704.07809

[12] Nisha B., Soorej Kamal, Jith P.S., Nayana S., M.S. Sinith, and Kiran Surendran. 2012. Sign Language Recognition using Support Vector Machine. 2012 International Conference on Advances in Computing and Communications, 122-125. DOI: http://doi.ieeecomputersociety.org/10.1109/ICACC.2012.27

[13] Alex Pentland and Thad Starner. 1996. Real-time American Sign Language recognition from video using hidden Markov models. Proceedings of International Symposium on Computer Vision - ISCV.

[14] Jinxu Tao, Wenwen Yang, and Zhongfu Ye. 2016. Continuous Sign Language Recognition using Level Building based on Fast Hidden Markov Model. Pattern Recognition Letters, 78 (April 2016), 28-35