How to Cite:
Damdoo, R., & Gupta, A. (2022). Gesture controlled interaction using hand pose model. *International Journal of Health Sciences*, *6*(S1), 10417–10427. https://doi.org/10.53730/ijhs.v6nS1.7493

Gesture controlled interaction using hand pose model

Rina Damdoo
Department of Computer Science & Engineering, Shri Ramdeobaba College of Engineering & Management, Nagpur, Maharashtra, India
Corresponding author email: damdoor@rknec.edu

Ashutosh Gupta
Department of Computer Science & Engineering, Shri Ramdeobaba College of Engineering & Management, Nagpur, Maharashtra, India
Email: guptaar_5@rknec.edu

***Abstract*---**Direct use of the hand as an input device is a smart method for providing natural human-computer interaction (HCI). Hand pose estimation is an attractive topic for research in recent years. It has been widely used in virtual reality. The domain of computer vision-based human hand three-dimensional shape and hand pose estimation has fascinated momentous attention recently due to its key role in various applications, such as natural human-computer interactions. Hand pose estimation is difficult due to some challenges. First, we need to detect the human hand which is very changeable. Second, the high degree of freedom leads to difficulties in pose estimation. In this paper, we aim to build a hand pose estimation system that can correctly detect a human hand and estimate its pose which can be useful in the areas like industrial automation, sign language recognition etc. We integrated a hand pose model with a game named U3D with 6 different gestures. As the object detection methods perform poorly in the palm detection tasks we used 21 hand joints and increased accuracy to approximately 95.63%. We used the Blender 3D computer graphics software toolset for creating animation, visual effects, motion graphics, and interactive unity 3D games.

***Keywords*---**Human-Computer Interactions, Hand pose estimation, Hand Pose Model, Blaze palm detection.

Introduction

A 3D model integration with human physical input refers to the ability to embed a person at different locations into a shared virtual environment. In such environments, it is essential to provide users with a credible sense of 3D tele-

Manuscript submitted: 27 March 2022, Manuscript revised: 18 April 2022, Accepted for publication: 9 May 2022

presence and interaction capabilities. This article presents a full real-time 3D-modeling system that uses hand gestures to control the person/character. The system integrates 3D models with human physical input. A 3D-modeling system with human physical input generally has three major parts:

1) Gesture/pose Recognition System
2) Game development
3) Integration of Gesture/pose Recognition System and Game

Gesture/pose Recognition

In palm detection, the color of skin, background subtraction, hand image extraction, edge detection and histogram analysis are used to achieve the goal. Followed by pose estimation by extracting features and classifying them into one of the categories.

Game development
It is an understanding of what is involved in building Virtual Reality games for Unity.

Integration of Gesture/pose Recognition System and Game: Integration is all about basic body-as-input interaction example where real time results from a hand tracking model (web cam stream as input) is mapped to the controls of a web-based game. The integration should be fairly accurate. So, light weight hand detection model can be used to track player hands and enable real time body-as-input interactions.

In this work, web camera has been used to capture gestures provided by the user, that are recognized and classified as one of the six gestures by our system. Accordingly it can be observed that the character in the U3D game performs the actions. We have predefined six hand gestures, but in future our work can be extended to classify many gestures. Fig. 1 shows Proposed U3D Game using Hand Pose Model. In section 2 we review work related to gesture recognition. Section 3 describes proposed approach towards gesture recognition using hand pose estimation followed by results and conclusion in sections 4 and 5 respectively.

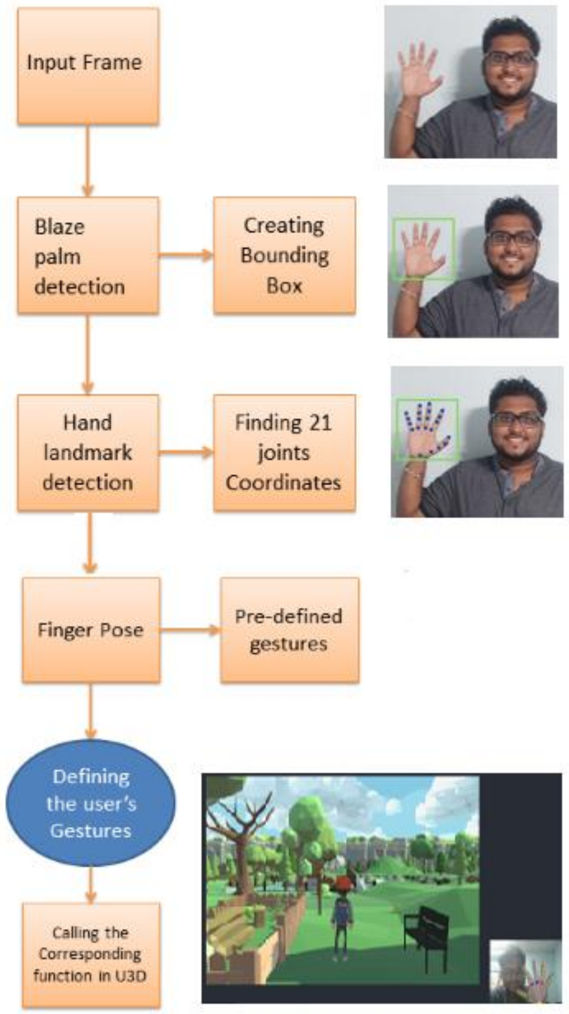


Fig. 1: Proposed U3D Game using Hand Pose Model

Literature Review

L. Ge et.al. [1], using multi-view CNNs propose a 3D regression method that can better exploit depth cues to recover fully 3D information of hand joints without model fitting and generated a set of heat-maps of multiple views from the multi-view CNNs and fused them together to estimate 3D hand joint locations.

Zhang et.al. [2], recommend a network SE-Resblock. Their model has divided into two parts: 2D hand joints estimation and depth coordinate estimation. They propose a new method that utilizes the information between adjacent joints on one finger to get accurate 3D hand pose estimation. Because of the self-occlusion of the hand, the depth information of hand joints is difficult to be extracted directly. To overcome this problem, they first regress the depth of a part of hand joints, then it is exploited as a reference, and concatenated the features and the output of the 2D joints estimation module for the next stage to estimate the depth of adjacent joints.

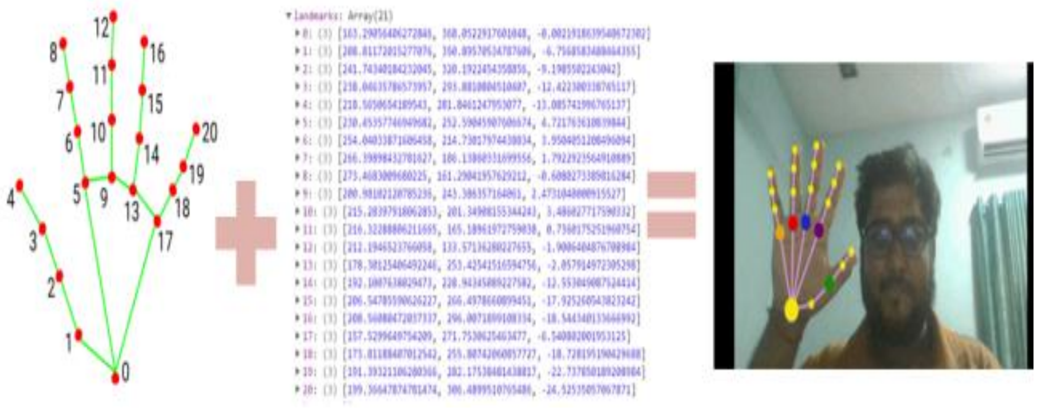
L. Ge et.al. [3], present a 3D CNN-based hand pose estimation method. By adopting the projective D-TSDF, they encoded the hand depth image as a 3D volumetric representation which is then fed into the 3D CNN and show that the 3D CNN mapping the 3D volumes to 3D joint locations in a single pass is easy to be trained in an end-to-end manner. P. Vicente et.al. [4], present a method for pose estimation of a robotic hand, based on a particle filter and GPU framework. Lin Huang [5], present a comprehensive survey of state-of-the-art 3D hand shape and pose estimation approaches using RGB-D cameras. Related RGB-D cameras, hand datasets, and performance analysis are also discussed to provide a holistic view of recent achievements.

Chen et.al. [6], designed a model called the spherical part model (SPM) and train a deep convolutional neural network using the model to estimate hand pose based on prior knowledge of the human hand. Many other researchers have worked on different techniques like hand pose trajectory tracking [7, 8], in various applications like Automotive Interfaces [9, 11] and sign recognition [10].

Proposed Work

Proposed work processes input video from the user webcam and each frame is send to the Blaze palm detection model, which returns hand bounding box around the palm with the coordinates of bottom left corner and top right corner. The frame is then cropped, thus containing only the image with palm cropped with reference to bounding box. This frame is given as input to Hand Landmark detection model to detect 21 3D points on the palm (Fig. 2).

We used Finger Pose Library to estimate the hand gesture input from the user. Finger pose library uses the hand landmarks detected by TensorFlow's hand pose model to detect hand gestures like “Thumbs Down”, “Thumbs Up” etc. There are mainly 3 Steps for detecting the Gesture.



```
▾Landmarks: Array(21)
 ▸0: (3) [163.29056406272845, 368.0522917601048, -0.0021918639540672302]
 ▸1: (3) [208.81172015277076, 350.8957053478760б, -6.756858348084355]
 ▸2: (3) [241.7434018423204б, 320.1922454350856, -9.198550224304б2]
 ▸3: (3) [238.04635786573957, 293.8810804510687, -12.422300318765117]
 ▸4: (3) [218.50506541895Ӏ3, 281.8461247953077, -13.085741996765137]
 ▸5: (3) [230.45357746049682, 252.5904590760б674, 4.721763610839844]
 ▸6: (3) [254.0403387160б458, 214.7301797443083Ӏ, 1.9504051208496094]
 ▸7: (3) [266.3989843278162, 186.1380831699556, 1.7922923564910889]
 ▸8: (3) [273.4683009680225, 161.2904195762921, -0.6088273385816284]
 ▸9: (3) [200.98182120785236, 243.3863571640б1, 2.4731048000915527]
 ▸10: (3) [215.28397918062853, 201.349081553442Ӏ1, 3.4860277175903З2]
 ▸11: (3) [216.3228880621165, 165.1896197275983Ӏ, 0.7360175251960754]
 ▸12: (3) [212.19465237660Ӏ8, 133.5713628027655, -1.900640487670898Ӏ]
 ▸13: (3) [178.3812540649224б, 253.4254151659475б, -2.0579149723052Ӏ8]
 ▸14: (3) [192.1007638029473, 228.9434588922758Ӏ, -12.553049087524414]
 ▸15: (3) [206.5478559062б227, 266.4978660099451, -17.925260543823242]
 ▸16: (3) [208.5608847203733Ӏ, 296.0071809108334, -18.544340133666992]
 ▸17: (3) [157.52996407542Ӏ9, 271.7530625463477, -6.540882001953125]
 ▸18: (3) [173.0118848701254Ӏ, 255.8874206085772Ӏ, -18.728195190429688]
 ▸19: (3) [191.3932110628036б, 282.1753840143881Ӏ, -22.737850189208984]
 ▸20: (3) [199.3664787478147Ӏ, 300.4899510765486, -24.525350570б7871]
```

Fig. 2: Hand Landmark detection model to detect 21 3D points on the palm

Analyzing the User's Gestures

In this step, the direction and curl of each individual finger is estimated. Using the coordinates provided by the hand pose model, it basically calculates the angle between our joints to predict the curl of each finger and the coordinates also provide the direction in which the finger is pointing towards. Figure 3a shows the index finger pointing toward the RIGHT and it is half curled. The amount of curl can be determined using the coordinates of each joint provided by the landmark model. Fig. 3b, 3c, 3d show the images for an example of how the index finger is curled, No curl, Half curl, and Full curl fingers respectively.

Setting up Gestures

One needs to represent the gestures that are expected from the user, by setting the direction, curl, and confidence for each finger. Table 1 lists the set of directions we used in this work.

Comparing the Confidence

'Confidence' is a number between 0 and 10 which describes how accurately a given combination of finger curl/positions matches a pre-defined gesture. One should design gestures so a perfect match will cause a confidence value of 10. Ten Finger Pose library is used to classify the computed landmark configuration into a discrete set of gestures. The model uses accumulated angles of joints to determine the state of the fingers and is then mapped to a set of pre-defined gestures. If the confidence value calculated is greater than or equal to 70%, then it is classified into one of the pre-defined gestures and this classification is then sent to the unity instance for further action in the unity environment.

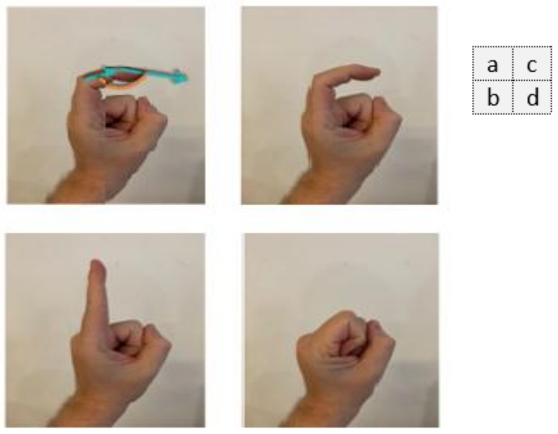


Figure 3: a) Index finger pointing toward the RIGHT, b) No curled index finger, c) Half curled index finger, d) Full curled index finger

Table 1
Set of the finger directions

Finger Direction	**Name**
0	Vertical Up
1	Vertical Down
2	Horizontal Left
3	Horizontal Right
4	Diagonal Up Right
5	Diagonal Up Left
6	Diagonal Down Right
7	Diagonal Down Left

Drawing a virtual hand

We draw a virtual hand across the webcam image by creating a canvas across the video feed and drawing the Arcs and the Strokes across it. To draw it, we used coordinates in the landmarks Array provided by the hand pose Model.Fig. 4 shows a detailed data flow diagram of our gesture detection system.

Blender

In this work, free and open-source 3D computer graphics software toolset Blender has been used for creating animation, visual effects, motion graphics, and interactive unity 3D games. There are several steps to create a 3D character in Blender. These take us from constructing a basic framework that will be the foundation for everything such as Preparing Drawings, Inserting Simple Shapes, Using Layers, Texturing Model, Rigging Model for Animation, and Rendering characters.

Unity 3D

Users can create games and experiences in both 2D and 3D with Unity, which includes a Scripting API and drag-and-drop functionality as well as plugins for use with the Unity Editor and games themselves. Unity provides a selection of tools that let creation of environmental features such as landforms and vegetation.

Environment

To make the environment, Unity's drag and drop functionality are used. Various objects are dragged, including cliff structures, terrain models, tree models, rocks, and man-made structures. All objects and models on the U3D game map have rigid collider parameters, which prevent the character from passing through them.

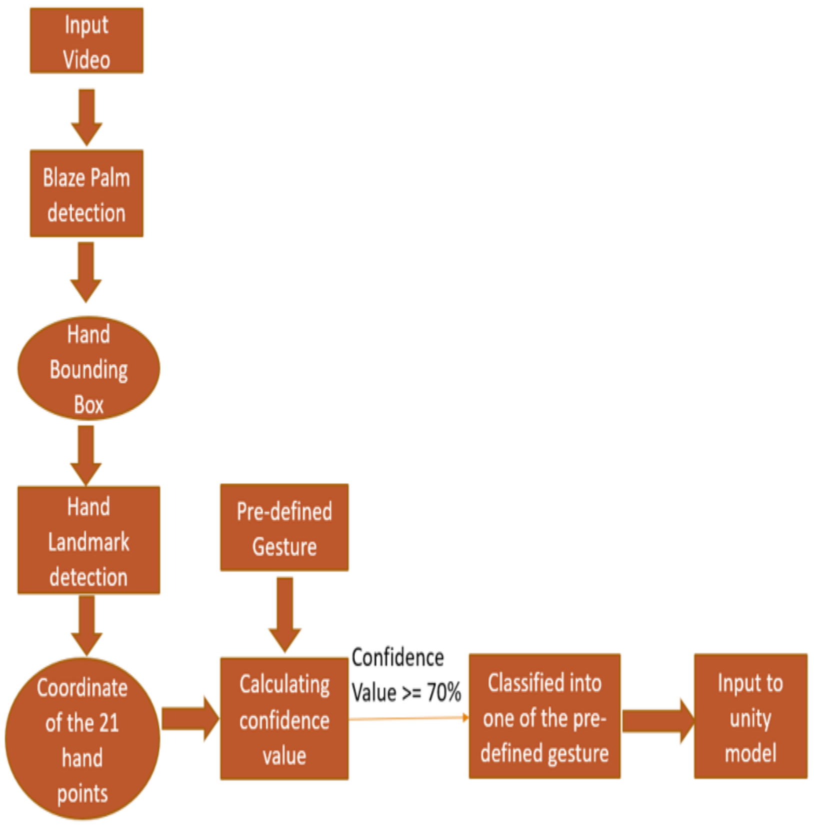


Fig. 4: Data Flow Diagram of Gesture Detection system

Character Movement and Animation

The character was defined to move freely in the built environment. For this movement, we predefined 6 controls, which are:

1. Stop
2. Crouch / Down
3. Jump
4. Rotate Left
5. Rotate Right
6. Move forward

To implement all these movements with ease, the character is given various animations, which are:

1. Idle Breathing
2. Stationary Character Crouch/ Roll
3. Stationary Character Jump
4. Camera Move Left
5. Camera Move Right
6. Running Character Crouch/ Roll

These animations are triggered at specific commands, by the use of Animation State graph. This graph defines various states of animation, and all the valid transitions from one animation to other.

Integration

The Integration of the Gesture Recognition System and the Unity Model is done in 3 steps:

1. Convert the Unity Model to HTML.
2. Send input data generated from Gesture Recognition System to the Unity Model.
3. Call corresponding functions of Unity Model based upon the input.

Step 1: Convert the Unity Model to HTML
Using WebGL, the unity model is converted into HTML, and the model can then be loaded in a browser by opening the html document. Therefore, 3D graphics can be displayed in real-time without plug-ins. In addition to OpenGL, WebGL can be combined with other HTML elements or composited with any part of the page.

Step 2: Send input data generated from Gesture Recognition System to the Unity Model.
An HTML file is the link between the gesture recognition model and the Unity model. Therefore, we can send data input to the html file and the unity model can extract data from it.

Step 3: Call corresponding functions of Unity Model based upon the input. To send data from React.JS to C# “SendMessage()” can be used.

Results and Discussions

Hand Pose model, which uses the blaze palm detection to classify gestures, is more accurate than a conventional object detection model. It is a relatively lightweight package consisting of approximately 12MB weights, making it suitable for real-time inference. Table 3 presents the results collected for 1000 sample frames for each of the six gestures as shown in Fig. 5. Table 4 presents the confusion matrix. Authors claim that test data contains images with different orientations and occlusions for the specific gestures. In future we will try to incorporate illumination variations in the same. Hand Pose model show percentage accuracy of about 95.63% in gesture classification using a regular cross entropy loss.

Because of the self-occlusion of hand, the joint information of hand joints is difficult to be extracted directly. Thus it can be observed from Table 3 that percentage accuracy is less for the gestures Move Down, Move Up, Run. Also it has been observed that the overall performance is independent of duration of gesture but depends on the speed with which the gesture is performed.

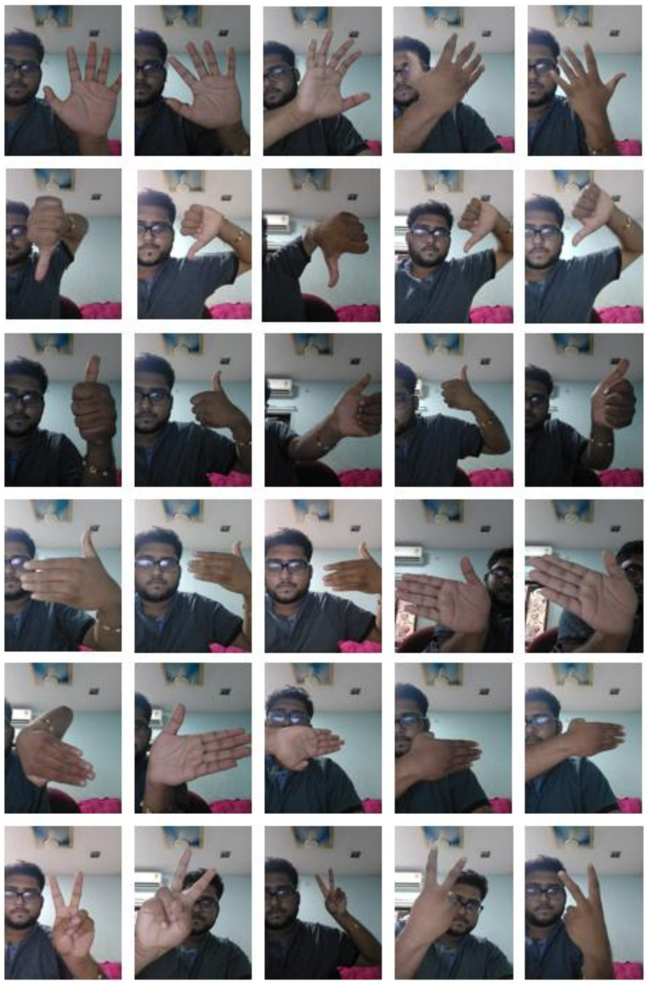
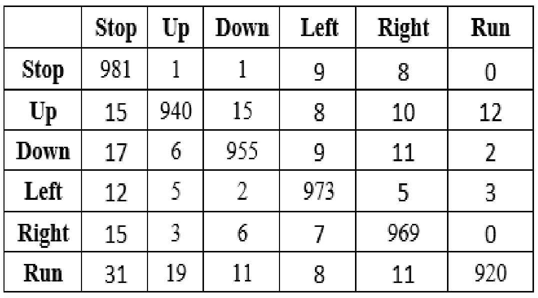


Fig. 5: Sample images for various Gestures Stop, Move Down, Move Up, Move left, Move Right, Run

Table 3
Results Table

Gesture Class	#Correct Gesture Recognition out of 1000 samples	Average Confidence Value of Correctly identified Gestures	% Accuracy of Gesture Recognition
Stop	981	9.4	98.1
Up	940	8.8	94.0
Down	955	8.5	95.5
Left	973	9.3	97.3
Right	969	9.3	96.9
Run	920	8.4	92.0

Table 4
Confusion Matrix



	Stop	Up	Down	Left	Right	Run
Stop	981	1	1	9	8	0
Up	15	940	15	8	10	12
Down	17	6	955	9	11	2
Left	12	5	2	973	5	3
Right	15	3	6	7	969	0
Run	31	19	11	8	11	920

Conclusions

Hand gestures as input is a smart method for providing natural human-computer interaction. So, Hand pose estimation is found to be a hot topic for research in recent years. Through this work, we tried to examine Hand pose estimation by applying it to gesture-controlled games and creating an Avatar.

Acknowledgements

We are grateful to all the authors in the reference section, who gave us a way to move in that direction. They all had used enormous methods for remote sensing image scene classification.

References

1. L. Ge, H. Liang, J. Yuan and D. Thalmann, "Robust 3D Hand Pose Estimation From Single Depth Images Using Multi-View CNNs," in IEEE Transactions on Image Processing, vol. 27, no. 9, pp. 4422-4436, Sept. 2018, doi: 10.1109/TIP.2018.2834824.
2. Zhang, Xiongquan & Huang, Shiliang & Ye, Zhongfu. (2021). Accurate 3D hand pose estimation network utilizing joints information. Signal Processing: Image Communication. 90. 116035. 10.1016/j.image.2020.116035.
3. L. Ge, H. Liang, J. Yuan and D. Thalmann, "Real-Time 3D Hand Pose Estimation with 3D Convolutional Neural Networks," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 41, no. 4, pp. 956-970, 1 April 2019, doi: 10.1109/TPAMI.2018.2827052.
4. P. Vicente, R. Ferreira, L. Jamone and A. Bernardino, "GPU-Enabled Particle Based Optimization for Robotic-Hand Pose Estimation and Self-Calibration," 2015 IEEE International Conference on Autonomous Robot Systems and Competitions, 2015, pp. 3-8, doi: 10.1109/ICARSC.2015.25.
5. Lin Huang, Boshen Zhang, Zhilin Guo, Yang Xiao, Zhiguo Cao, Junsong Yuan, Survey on depth and RGB image-based 3D hand shape and pose estimation, Virtual Reality & Intelligent Hardware, Volume 3, Issue 3, 2021, Pages 207-234, ISSN 2096-5796, https://doi.org/10.1016/j.vrih.2021.05.002.
6. Chen, Tzu-Yang, Pai-Wen Ting, Min-Yu Wu and Li-Chen Fu. “Learning a deep network with spherical part model for 3D hand pose estimation.” ICRA (2017).
7. Yuan-Hsiang Chang, Chen-Ming Chang, "Automatic Hand-Pose Trajectory Tracking System Using Video Sequences", INTECH, pp. 132- 152, Croatia, 2010
8. Erik B. Sudderth, Michael I. Mandel, William T. Freeman, Alan S. Willsky, "Visual Hand Tracking Using Nonparametric Belief Propagation", MIT Laboratory For Information & Decision Systems Technical Report P2603, Presented at IEEE CVPR Workshop On Generative Model-Based Vision, Pp. 1-9, 2004
9. Eshed Ohn-Bar, Mohan Manubhai Trivedi, "Hand Gesture Recognition In Real-Time For Automotive Interfaces," IEEE Transactions on Intelligent Transportation Systems, VOL. 15, NO. 6, December 2014, pp 2368-2377
10. Ruiduo Yang, Sudeep Sarkar, "Coupled grouping and matching for sign and gesture recognition", Computer Vision and Image Understanding, Elsevier, 2008
11. Rina Damdoo, “N-Gram based Smart Living Machines (SLM) on IOT Platform”, International Journal of Innovative Technology and Exploring Engineering (IJITEE), Volume8, Issue-8S3, pp 293-300