Real-Time Motion Control of a Humanoid Robot Using Deep Learning

To cite this article: A S Faraz Ahmed *et al* 2021 *J. Phys.: Conf. Ser.* **2115** 012007

Real-Time Motion Control of a Humanoid Robot Using Deep Learning

A S Faraz Ahmed[1,*], V Sudharsan[2], Arockia Selvakumar Arockia Doss[3]

[1] Department of Information Technology, Panimalar Institute of Technology, Chennai, Tamil Nadu, India.

[2] Assistant Professor, Department of Electrical and Electronics Engineering, Panimalar Institute of Technology, Chennai, Tamil Nadu, India.

[3] Senior Associate Professor, Design and Automation Research Group, SMEC, Vellore Institute of Technology, Chennai, Tamil Nadu, India.

[*]farazahmed3071@gmail.com

Abstract: This paper discusses the research work done for controlling the humanoid robot manually using deep learning. For teaching, personal assistance, search and rescue humanoid robot are used. Controlling manually makes it to do any task without any explicitly programming. Existing technique for manually controlling the humanoid are heavily dependent on hardware and they are not cost efficient. This paper proposes a novel method for controlling the humanoid using a 2D camera. The image from the 2D camera is processed and skeleton of the human body is captured using deep learning. Then the skeleton is used to control the actuators present in the humanoid robot using image classifier and ROS. As a proof of concept the upper body of the humanoid robot is controlled in real time using this method.

Keywords: deep learning – real time motion control - humanoid robot - ROS

1. Introduction

A humanoid robot is a highly complicated two legged robot capable of doing any task assigned to it. Conventionally, a humanoid robot is designed and developed for a particular application like personal assistance, search and rescue, manufacturing, etc. But they are bad at multitasking like; a search and rescue robot can't work in a manufacturing plant even though they have physical capabilities. The search and rescue robot need to be programmed explicitly to work in a manufacturing plant and same apply for other application robots. To solve this problem a person can manually control the humanoid robot and no modification in software is required. For manual control sensors, jogging tool and game controllers are used [1]. Most tools work by capturing the motion of the human body by constructing the skeleton using sensors [2][3]. The major problem with these tools is hardware is very costly and not suitable for regular use. Most controlling tools need to be worn in hands and legs which create discomfort for the end user. They are not user friendly. A more efficient way of capturing the motion is using a 2D camera for skeleton construction. Pose estimation is a computer vision technique for mapping the human body joints and using that skeleton is extracted. There are many pre-trained pose estimation models available in open source. But, pose estimation models from Tensorflow built using deep learning shows more advantage than all other models. So, it is used for constructing the skeleton from an image. In order to make the robot perform a specific pose or movement a class with a pose name is created containing the image of that pose. A class is a collection of similar image with some change in parameter like image size, colour contrast, etc. Totally 7 classes are created denoting 7 pose the humanoid can perform. The created image classes are fed for training to an image classifier. The image classifier identifies the name of the class and passes it to ROS (Robot Operating System) where the pose is achieved. A comparative study is made for evaluating the performance of this approach. For comparative study, raw image from the camera and skeleton extracted image are used in a classifier. Raw image is a image obtained from camera without any preprocessing. Using the comparative study results the raw image or skeleton extracted image is chosen for robot control.

2. Architecture

The general architecture of the manual control of humanoid robot is a person used to have controllers mounted in his body suit or hold it in his hands. A computer used to tract the location of the controllers and creates a virtual skeleton in the 3D space. Using that skeletons endpoint location inverse kinematics finds the respective angle for each joint. This angle is passed to the robot for control.

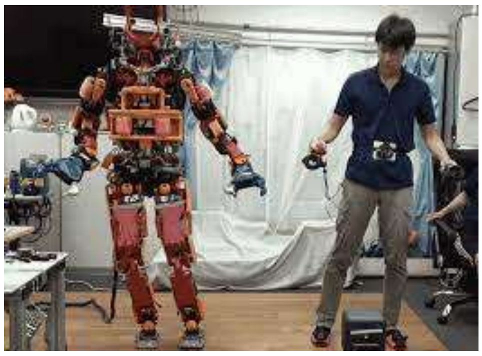


Figure 1. Controlling a life sized humanoid robot using VR game controllers.

The architecture of the presented novel approach is given in figure 2. An image from the camera is passed to the classifier and the name of the image class is passed to the ROS where it has a dictionary of class names and its respective servo values for making the pose.

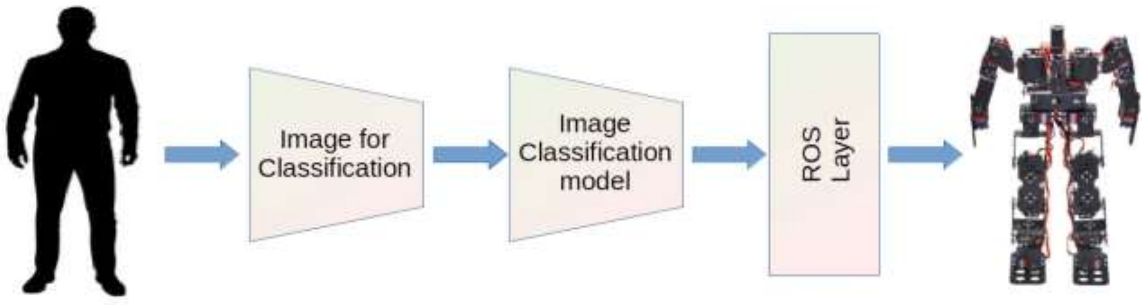


Figure 2. Proposed Novel Approach

An image of a person is obtained from a camera and passed for classification. The name of the pose is identified by the image classifier and passed to the ROS layer for robot control. The image from the camera and robot pose will be same.

3. Methods

The image captured using camera is passed to each layer. The layers are in sequential order where raw image at the starting and robot movement at the last.

3.1. Image for classification

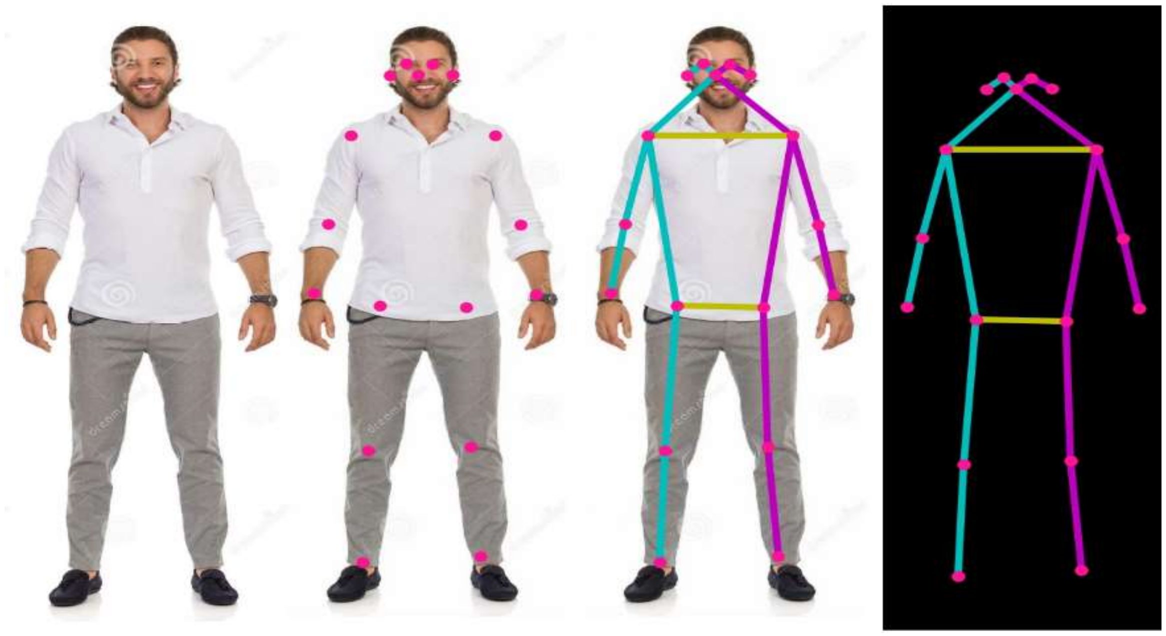
The image passed to the classifier has two categories. They are raw image from camera and skeleton extracted image. By evaluating the performance of both the categories in the classification best performing image is used.

3.1.1 Raw image
The raw image is an image obtained from the sensors of the camera. Most image taken using smartphones, DSLR camera are processed image. The algorithm for processing the image used to modify the colour contrast, blur, and other channels of the image for enriching the quality and viewing experience. The processing algorithm is not common and varies for every brand and camera of type. Using a processed image for classification may result in change of accuracy. This is not suitable for evaluating the performance of classifier. Hence the raw image is used.
3.1.2 Skeleton extracted image
For extracting the skeleton pose estimation technique is used. Pose Estimation is a computer vision technique of mapping the location of keypoints in an image [4]. Keypoint is a joint or spatial location like shoulder, ankle, etc. All the keypoints are denoted using a dot (refer Figure 3). The dots are connected to construct the skeleton. The constructed skeleton is plotted in a plain image and it is used for classification.
3.2. Classification:
As the name suggest classification is a machine learning problem which identifies the pattern present between each class and classifies them. The classifier identifies the class by learning the features. A feature is an individual measurable property or phenomenon which is used for classification [4]. A deep learning classifier can consider anything as a feature and no one knows what feature it took. Classification is also used to classify images, audio, spam mail and even video.
3.3. ROS layer:
For controlling the humanoid robot ROS is used. ROS is a framework created for developing software for robots. It acts as a middle ware between hardware and software [8]. Without ROS any robot can be controlled. But it takes more time for development and hard to debug. ROS is like a abstract layer which reduces the complicity in writing code and easy to migrate to other development board independent of their architecture. It also reduces the line of code and has a logs file for easy debug.

4. Analysis

4.1. Image for classification
A video is recorded for each pose using a pi camera with varying duration. Each video was created with varying dress code, dress colour, and background. The video is then converted into frames or images and saved for classification. The varying video duration resulted in a imbalanced datasets. So the class which has minimum number of images is taken as a constant and remaining images were removed to make a balanced datasets.
4.1.1 Raw image
The raw image datasets is passed to the classifier for training and the classification model is tested for evaluation.
4.1.2 Skeleton extracted image
The images in raw image datasets are used to create skeleton image datasets. For skeleton extraction a pre-trained pose estimation model called MoveNet from TensorFlow is used [4]. The MoveNet offers two pre-trained model. They are MoveNet Lighting and Movenet Thunder. The MoveNet Lighting is faster with less accuracy and MoveNet Thunder is slower compared to lighting and has more accuracy. Both the pre-trained models are built using CNN architecture and trained using COCO (common object in content) datasets [5]. For real time motion control fast working model is required. So, MoveNet lighting is used. The model follows single person pose estimation approach for keypoint detection with a accuracy of 0.0 to 1.0.

The pose estimation model has a list of part ID for each spatial location (body part) and assigns each part ID to a location in a image denoting the part location. For example nose has a part ID of 0 and it is assigned to a location 80, 40 where 80 is the x-axis and 40 is the y-axis of an image. By this, all 17 keypoints are detected. The detected key points from the pose detection model will be in a sequential order. The order will be as follows: nose, left eye, right eye, left ear, right ear, left shoulder, right shoulder, left elbow, right elbow, left wrist, right wrist, left hip, right hip, left knee, right knee, left ankle, right ankle. Two keypoints from the above list is considered as a pair. Different colour line connects the pair. Totally 18 pair forms the skeleton. The paired points and lines are plotted in a black background image of raw image width and height [6][7].



(a) Raw Image (b) Keypoints Identification (c) keypoints with skeleton (d) Extracted skeleton

Figure 3. Skeleton extraction

4.2. Classification model:

A deep learning image classification model is built and trained using 7 classes where each class represents a pose. Each class has approximately 2400 images and it is split into 80% as training datasets and 20% as validate datasets. The classifier model was built from 2 to 7 classes and a new class is added after noting certain parameters like model training time, prediction percentage, etc. The above mentioned steps are followed for both raw image datasets and skeleton image datasets.

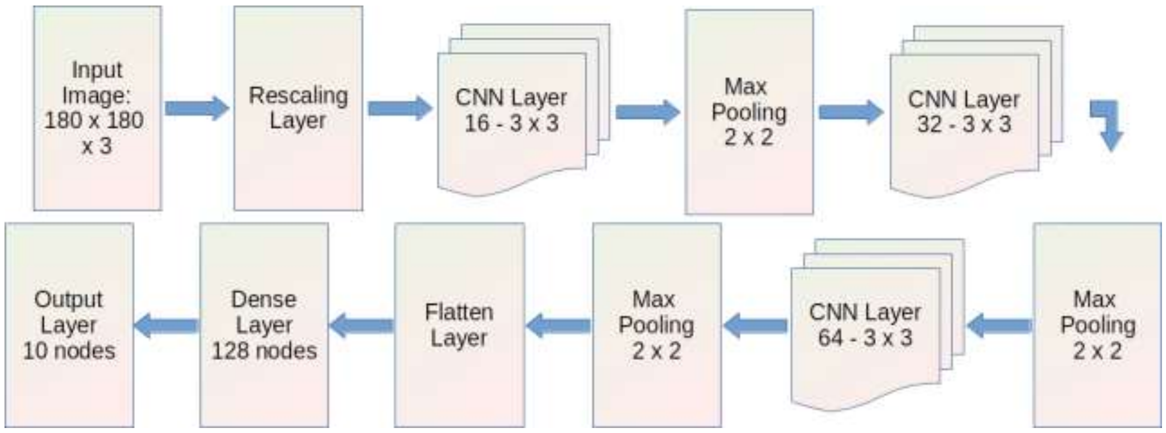


Figure 4. Pipeline of classification model

A convolutional neural network is used to build the classifier model instead of a linear classifier, because the CNN can predict with more accuracy compared to other classifier [8]. The model uses a rescaling layer, three convolution layers next to each other with a max pool layer in each of them, a flatten layer followed by dense layer and an output layer. The rescaling layer rescales the input image of [0, 255] to [0, 1] range. The convolution layers extract the features from the image but with the change in position or orientation of the feature results in different feature maps. To overcome this problem max pool layer is used next to each convolution layer. The max pool layer calculates the maximum or largest value in each patch of each feature map and also reduces the chances of over-fitting. Over-fitting is a stage where model runs too good in the train datasets but fails at the test datasets. The flatten layer is used to reforms the input image. The dense layer has a fully connected 128 nodes or neurons. The convolutional layers and dense layer is activated by ReLU activation function except for the output layer. And the last layer returns the confidence of each class to which the image belongs to.

4.3. ROS layer:
The ROS is a cross platform framework and support major programming languages like Java, C++ and python. Python and C++ are used for programming. Deep learning uses python and robot uses C++. Both languages communicate to each other using ROS node. The hardware of humanoid robot consists of 18 metal gear RC servo which forms 18 DoF (Degree of Freedom) and uses two Arduino Uno for control [9][10]. The two Arduino uno utilizes I2C protocol for communication and one acts as master and another acts as slave. All the boards and electronics including servo are powered by a 2 cell Li-Po battery. In the ROS layer, the publisher node is integrated with the classification model to publish data. The Arduino Subscriber node receives the data from the Publisher node through ROS Core and controls the robot [11]. The publisher runs on Nvidia Jetson Nano computer and Arduino Subscriber runs on Master Arduino Uno. The connection is made between the development board and computer using USB cable.

5. Results

5.1. Image for classification
Initially the datasets had 9100 images that is 1300 image for each pose. But the number of images was not sufficient to make a correct prediction and resulted in a under-fit of the classifier. So more images were added, and the final datasets had 16800 images, where each class has exactly 2400 images. The skeleton image datasets was created using a 18 keypoint pose estimation model built from caffee framework [12]. But, the time it took for single image pose estimation is approximately 17 seconds. So it is replaced with TensorFlow's MoveNet - Lightning pose estimation model and some preprocessing is done to speed-up the training time. The preprocessing in the raw image datasets resulted in an under-fit so it is not done.

5.2. Classification model:
The noted values for classification model using raw image datasets and skeleton extracted datasets shows that the skeleton extracted model has more benefits than raw image model. Skeleton extracted model has more accuracy compared to raw image model and has less training time. In raw image model the class 4 to 7 has attained the underfit and its accuracy is given in the below table. That is the classifier has predicted the image wrongly and the accuracy of the prediction is taken for comparison.

Table 1: Raw Image Model

Number of classes	Training Time	Prediction Value
2 Class: (up, down)	25 minutes	73.11 %
3 Class: (2 Class + full)	28 minutes	55.31 %
4 Class: (3 Class + right)	31 minutes	47.49 % + underfit
5 Class: (4 Class + left)	35 minutes	40.42 % + underfit
6 Class: (5 Class + rightup)	40 minutes	28.73 % + underfit
7 Class: (6 Class+ leftup)	49 minutes	25.22 % + underfit

Table 2: Skeleton Extracted Model

Number of classes	Training Time	Prediction Value
2 Class: (up, down)	2 minutes 25 seconds	76.21 %
3 Class: (2 Class + full)	4 minutes 14 seconds	57.61 %
4 Class: (3 Class + right)	4 minutes 32 seconds	47.54 %
5 Class: (4 Class + left)	7 minutes 18 seconds	40.46 %
6 Class: (5 Class + rightup)	7 minutes 33 seconds	35.22 %
7 Class: (6 Class+ leftup)	8 minutes 36 seconds	31.18 %

The Figure 5 shows the training time (t) of both the models. The average time difference for each class of skeleton image model falls less than 4 minutes and overall time it took stays less than 10 minutes. Whereas in raw image model average time is 29 minutes and overall time it took is 50 minutes. Both the models used same amount of images for training and raw image model training time is significantly high compared to skeleton extracted image. The reason of this difference is that the skeleton datasets used black background and using that cropping algorithms reduce the image size.

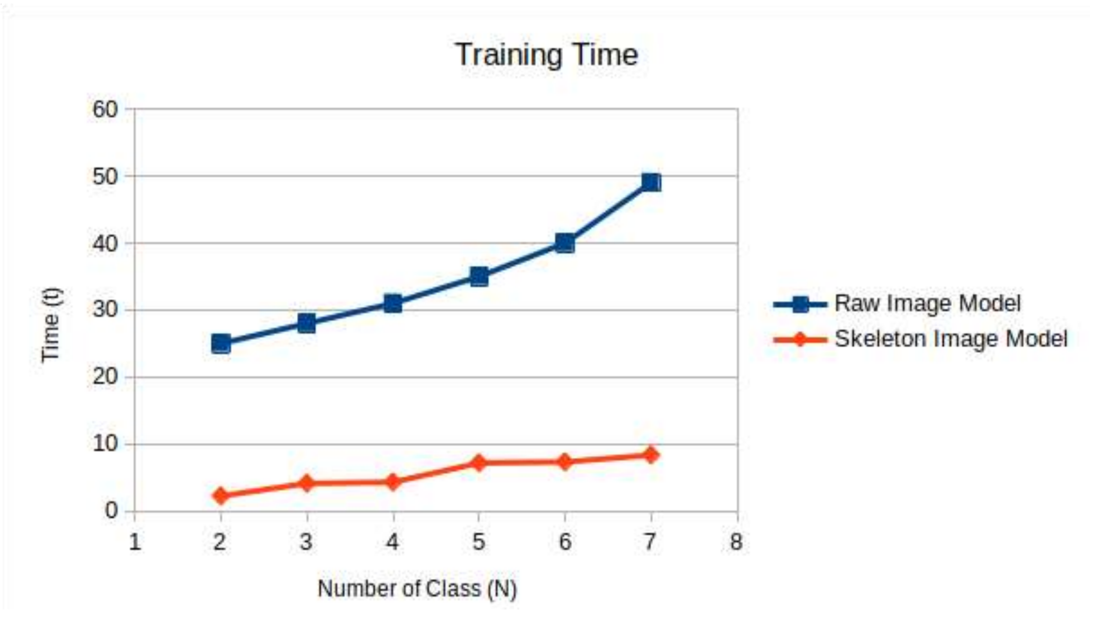


Figure 5. Training Time graph of classification

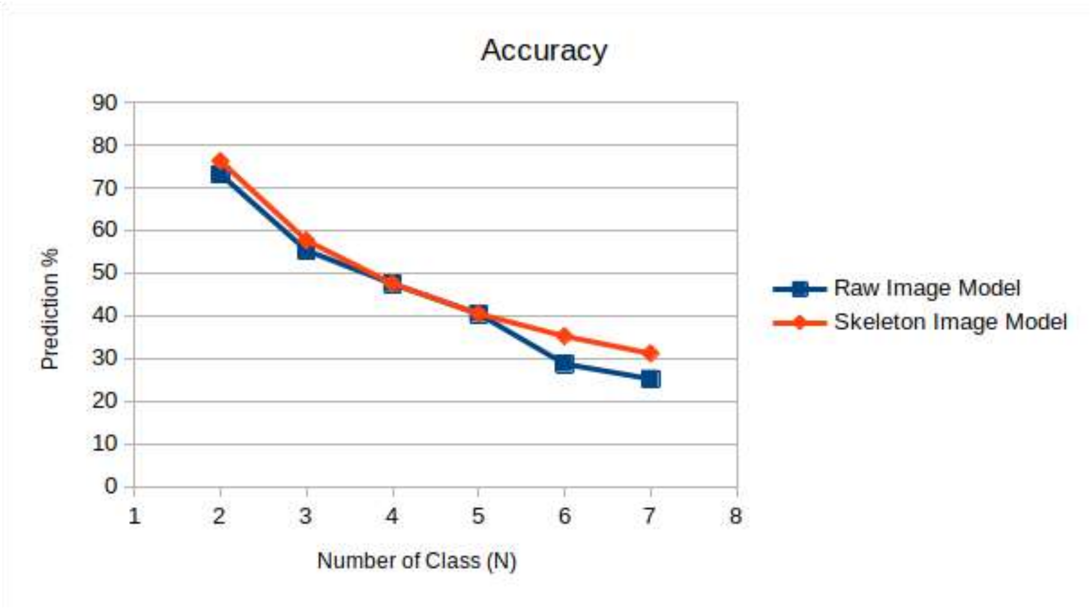


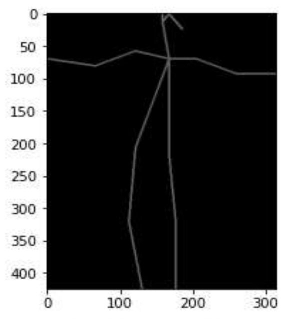
Figure 6. Accuracy graph of classification

The Accuracy graph (Figure 6) shows very less difference between the two models. The raw image model got under-fit for class 4 to class 7. This means the test image passed to the raw image model for class 4 to 7 predicted the image wrongly and that accuracy was taken for comparison. All the models of skeleton image model were tested with many test images and the model was able to predict the class correctly while maintaining a good accuracy. No over-fit or under-fit is detected in Skeleton extracted model. The test time is the time taken to converting a raw image to skeleton and passing it to a classifier for prediction. The test time for a single image running in a GPU is 0.989 ms and for CPU it is 1 second. On average a video for 1 second contains 24 frames, which means 24 images per second. The overall runtime should be less than 41ms to run it in real time. But in our case we got more 41 ms, this leads to real time control problem. To solve this problem frame skipping technique is used. Frame skipping is a process of skipping any number of frames in a 1 second video source. This increases the performance of the model and reduces computational power. Using this technique 1 frame is retrieved from every 24 frames and this helps in a smooth motion control in real time without any time delay more than 1.3 seconds. The poor performance of raw image model was due to it's complexity in image like random objects in the background, varying light intensity, etc. Normally this is not a problem for classifier because it takes more than 10,000 to 20,000 images for each class and this massive amount of images will help the classifier predict the class correctly with very high accuracy.

The performance of skeleton extracted model was not great in terms of accuracy though it never attained the under-fit or over-fit state. The reason why skeleton extracted model has not attained under-fit is that there is no random objects in background, change in skeleton colour intensity, etc. The skeleton extraction process made the feature easily identifiable and handle problems like blur image, colour contrast, varying camera viewpoint, bad image quality, etc. The common black background for train and test images in skeleton extracted model is also a reason for not getting under-fit. All classifier follows a rule of increase in number of images (n) results in more accuracy. The accuracy is directly proportional to the number of images. Using skeleton image for classification breaks the classifier rule. The main objective of this paper is to control the robot and not getting 100% accuracy in classification. The accuracy can be increased by adding more images to the datasets. For extreme testing of skeleton image model, new images were created using caffee pose estimation model and tested for evaluating the performance.

5.3.Test case 1

In skeleton extraction the thickness of the connecting lines is reduced to 2. This image is passed to the classifier and surprisingly the skeleton image model was able to find the class name correctly. There is no major change in accuracy.

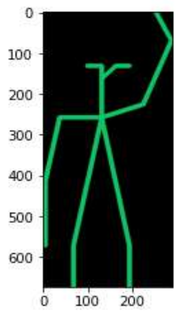


```
0.05324578285217285
This image most likely belongs to full with a 31.18 percent confidence.
```

Figure 7. Thin Line Skeleton

5.4.Test case 2

The skeleton extracted image with 10 line thickness is cropped manually and it is passed to the classifier. Again the skeleton image model was able to find the class name correctly with 2% fall in accuracy.

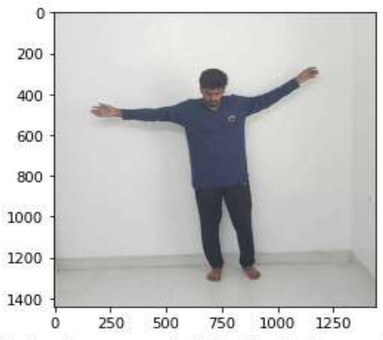


```
0.06931376457214355
This image most likely belongs to leftup with a 31.18 percent confidence.
```

Figure 8. Leg cropped with high thickness of skeleton

5.5.Test case 3

For final testing a raw image is passed to the skeleton image model. The model has never seen a raw image in its training or validation stage. So, while testing with a raw image the model was not able to identify the class and its accuracy was 31.18 %.

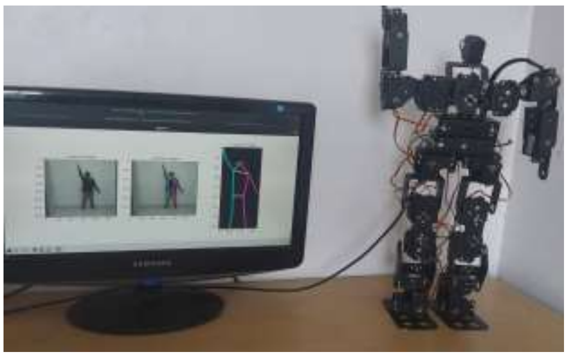


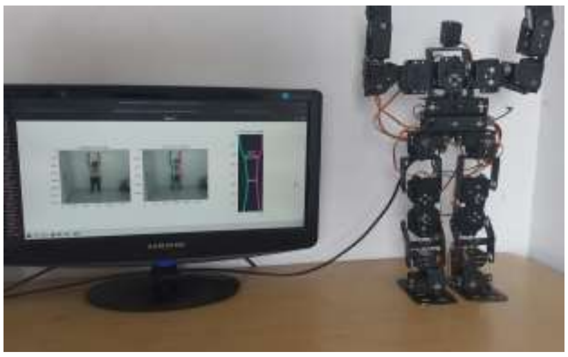
```
This image most likely belongs to up with a 31.18 percent confidence.
```

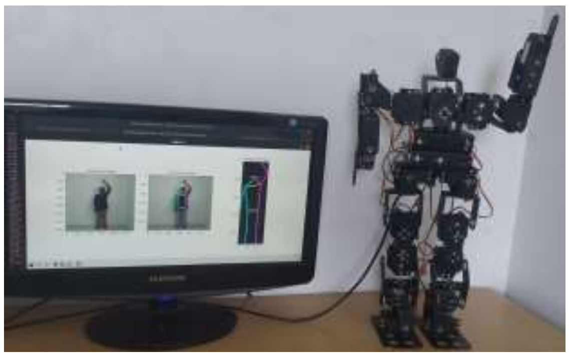
Figure 9. Raw image passed to the classifier

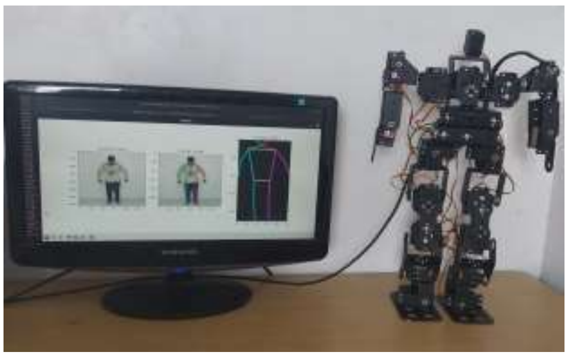
5.6. ROS Layer:

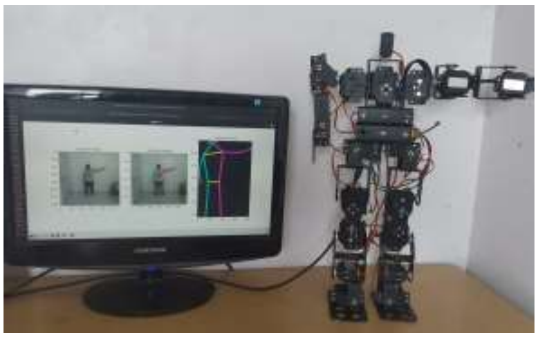
The skeleton image model is used for implementation. The image is captured using a pi camera and the skeleton is extracted from the image. The skeleton image is fed to the classifier and name of the pose is obtained. The pose name is passed to the ROS and the movement is achieved.

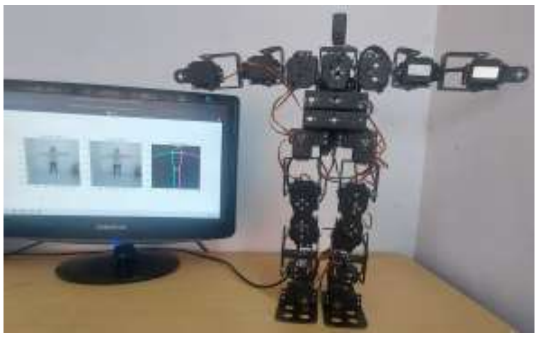

(a) Rightup


(b) Up


(c) Leftup


(d) Down


(e) Left


(f) Full

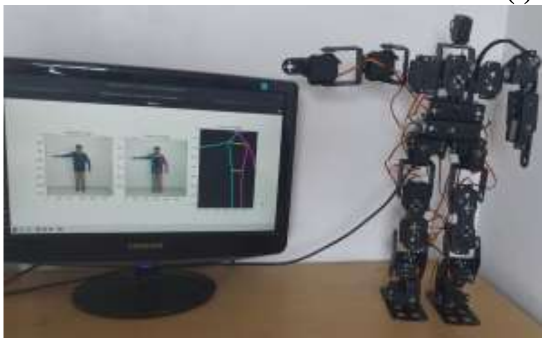

(g) Right

Figure 10. The pose is estimated, classified and the movement is achieved

6. Summary and future work

The capturing of motion using 2D camera shows more advantage over sensor. The drawbacks of sensors are it can react to external factors like heat, voltage, etc which changes the actual sensor value. Failure of device, malfunction is hard to detect in sensor. To overcome, these problems more than one sensor is used in place of a single sensor as a redundancy layer. Using more sensors leads to more computational time, more power usage; lesser free GPIO pins and increases the cost of prototype. A better solution will be using camera to control the humanoid. Camera can react to heat and voltage fluctuation but their range is very high compared to sensors. Using camera reduces GPIO consumption and creates room for other peripherals. Speaking of camera malfunction it is easily identified by using OpenCV's inbuilt module. The RIM & skeleton image models are tested in the humanoid in real time and skeleton image model shows more advantage over RIM. The application of this kind of pose control robot is that it can work in area where gesture or pose is required. A personal relation robot or teaching robot can also be controlled using this method. Even an exercise instructor robot can be placed in a quarantined zone and a human can teach the patients to do exercise.

The major advantage of this approach is that a low cost normal 2D camera can be used without compromising in the performance of the model or limitation in the movements, the skeleton image model can control other movable robots like wheeled robot, etc., this is due to the classifier in architecture. The sensors available for controlling the robot like Microsoft Kinect, IMU are very expensive compared to normal camera. The Kinect from Microsoft can detect two persons at a time. The IMU can give exact location of the joints in 3D space and cost increases with increase in number of joints to be tracked. Multiple 2D cameras can track a person in 3D space and cost is extremely low compared to other sensors and controllers. The skeleton image model uses classification to control the humanoid using pose estimation as a backbone. The same problem can be solved using regression. The regression approach gives raw integers as an output using image alone. For example brightness of an image, age of a person, etc.., But this method is hard to implement in real time because so much pre-processing need to be done before regression to get a usable data for every single image which increases the overall computation time. Future experimental study can give the pros and cons of regression in skeleton image model with lab tested data.

References

[1] Manon Kok, Jeroen D. Hol, Thomas B. Schön 2014 An optimization-based approach to human body motion capture using inertial sensors vol 47 3 79-85.
[2] Mohammed A. Hussein, Ahmed S. Ali, F.A. Elmisery and R. Mostafa 2014 Motion Control of Robot by using Kinect Sensor 1384-1388.
[3] Ishiguro, Yasuhiro and Kojima, Kunio and Sugai, Fumihito and Nozawa, Shunichi and Kakiuchi, Yohei and Okada, Kei and Inaba, Masayuki 2018 High Speed Whole Body Dynamic Motion Experiment with Real Time Master-Slave Humanoid Robot System 5835-5841.
[4] Alexander Toshev, Christian Szegedy 2014 DeepPose: Human Pose Estimation via Deep Neural Networks 1653-1660.
[5] Alexandru O. B˘alan, Leonid Sigal, Michael J. Black, James E. Davis, Horst W. Haussecker 2007 Detailed Human Shape and Pose from Images 10.1109/CVPR.2007.383340.
[6] Andrew G. Howard,Menglong Zhu, Bo Chen 2017 Mobilenets: Efficient convolutional neural networks for mobile vision applications arXiv:1704.04861v1.
[7] Zewei Ding, Pichao Wang, Philip O. Ogunbona, Wanqing Li 2017 Investigation of Different Skeleton Features for CNN-based 3D Action Recognition 617-622.
[8] Andrej Karpathy, George Toderici, Sanketh Shetty 2014 Large-scale Video Classification with Convolutional Neural Networks 1725-1732.
[9] Marc Bestmann, Norman Hendrich, Florens Wasserfall 2017 ROS for Humanoid Soccer Robots.
[10] Hsuan-Ming Feng, Ching-Chang Wong, Chih-Cheng Liu, Sheng-Ru Xiao 2018 ROS-Based Humanoid RobotPose Control System Design 4089-4093.
[11] Moonyoung Lee, Yujin Heo, Saihim ChoHyunsub Park, Jun-Ho Oh 2019 Motion Generation Interface of ROS to PODO Software Framework for Wheeled Humanoid Robot 456-461.
[12] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, Bernt Schiele1 2014 2D Human Pose Estimation: New Benchmark and State of the Art Analysis 3686-3693.