Saroj: Making NAO Dance

Bhavy Singh, Yasharth Bajpai, and Indranil Saha
Department of Computer Science & Engineering, Indian Institute of Technology Kanpur, Uttar Pradesh, 208016
Email: {bhavys, yashb, isaha}@cse.iitk.ac.in

***Abstract*—Robots have tremendous potential to be used for entertainment purposes. Humanoid robots are at the forefront of it as they have the capability of imitating human actions. In this project, we have developed a software system, Saroj, which is aimed at imitating dance moves as performed by a human in a given video. Our Software identifies and collects 2D keypoints [1] from a video and thereafter intelligently projects them into the 3D space [2]. These projections are then converted into dance moves and synthesized into scripts runnable by Nao. We have used the software Choreographe from Softbank Robotics to perform simulation of dances synthesized by Saroj for the Nao humanoid robot successfully.**

I. Introduction

Currently, the research and development of humanoid robots in its initial stages with attempts being made to get as close to a mechanical human as possible. Humanoids could theoretically perform any task a human being can, so long as they have the proper software. However, the complexity of doing so is immense. Initial aim of humanoid research was to build better orthosis and prosthesis for human beings however, it has now have diverged into several domains including military, entertainment and many more. Imitation learning is also a rising field of exploration in this regard wherein the robot learns activities seeing humans perform them and then imitates the activity on its own. There has been active research on humanoid robots imitating human physical movements however; the formalization of a 'Language for dance' isn't something that has been delved into. In this project, we take this effort forward by making the Humanoid, NAO, replicate the choreography as performed by a human in a video. Our problem can essentially be divided into two segments, the first one being extracting information from the video provided as an input to the software; and second being using the information to synthesize dance in the language subset(ref. section III-A).

II. Human Pose Estimation

Human pose estimation is one of the critical problems in computer vision that has been studied for well over 15 years. The reason for its importance is the abundance of applications that can benefit from such technology. It is defined as the problem of localization of human joints (also known as keypoints - elbows, wrists, etc.) in images or videos. It is also defined as the search for a specific pose in the space of all articulated poses. Strong articulations, small and barely visible joints, occlusions, clothing, and lighting changes make this a difficult problem. Saroj uses it for the capturing and reproducing choreography as formal dance notations. The estimation in three-Dimensional space is way more trickier than in the two-demsional setup as it requires those extra angles for the estimation of depth of each frame which are processed concurrently to give an estimate of each joint coordinate in the 3D space. As we accept an input as a two-dimensional video for wider usage, 3D keypoint extraction is not possible. Therefore, we use state of the art object detection algorithms to estimate the keypoints int three dimensions. In our software system, the process works in two phases, namely; Two Dimensional Keypoint extraction and Three Dimensional Pose Estimation from the two-dimensional data.

Fig. 1. Schematic representing the mapping of Image pixels to Human Body in 3D Space [1]

A. Two Dimensional Keypoint Extraction

On input of a video to Saroj, it pre-processes the video to a standard frame-rate of 50fps. Thereafter, two-dimensional location of Human joints are extracted from the input video using the techniques borrowed from Densepose [3], which aims at mapping all human pixels of an RGB image to the 3D surface of the human body, serving as a crucial step in solving our problem. Our Software system uses a tweaked implementation from Facebook AI Research's Detectron [1], for DensePose Estimation in COCO format. The implementation is written in Python and powered by the Caffe2 deep learning framework.

B. Three Dimensional Pose Estimation

Once the 2D keypoints have been extracted from each frame, they are mapped into 3D space using a fully convolutional model based on dilated temporal convolutions over 2D keypoints [2]. The model is trained over the Human 3.6 dataset using an additional feature of back-projection. The generated results are then forwarded to the dance synthesizer which processes them further.

III. Choreographic Language

A. Language Subset

Our Language closely follows from Labanotation using the basic movements for the left hand and right hand into movements

at 3 levels namely:

1. high = Above the shoulder level
2. mid = Parallel to shoulder level
3. low = Below the shoulder level

At each level there are 9 directional motions namely:
1. R = Right
2. L = Left
3. P = Place
4. B = Backward
5. LB = Left backward
6. RB = Right backward
7. F = Forward
8. LF = Left forward
9. RF = Right forward

B. Quantisation of Movements

The following are the values of the parameters for 4 out of the 54 total such hand movements.

LEFT HAND Joint Name and Motion Range

Joint name	Motion Range (degrees)
LShoulderPitch	Left shoulder joint front and back (Y)
LShoulderRoll	Left shoulder joint right and left (Z)
LElbowYaw	Left shoulder joint twist (X)
LElbowRoll	Left elbow joint (Z)
LWristYaw	Left wrist joint (X)
LHand	Left hand

LEFT HAND-HIGH-BACK Movements and Parameters

Joint name	Angle(radians)
LShoulderPitch	-1.74749
LShoulderRoll	0.0280893
LElbowYaw)	-1.56851
LElbowRoll	-0.776554
LWristYaw	0.00987219
LHand	0.0018125

LEFT HAND-MID-RIGHTBACK Movements and Parameters

Joint name	Angle(radians)
LShoulderPitch	-0.0223843
LShoulderRoll	-0.239475
LElbowYaw	0.0486595
LElbowRoll	-1.45994
LWristYaw	-1.30485
LHand	0.00273752

RIGHT HAND Joint Name and Motion Range

Joint name	Motion Range (degrees)
RShoulderPitch	Right shoulder joint front and back (Y)
RShoulderRoll	Right shoulder joint right and left (Z)
RElbowYaw	Right shoulder joint twist (X)
RElbowRoll	Right elbow joint (Z)
RWristYaw	Right wrist joint (X)
RHand	Right hand

RIGHT HAND-LOW-LEFT Movements and Parameters

Joint name	Angle(radians)
RShoulderPitch	0.59195
RShoulderRoll)	0.0280893
RElbowYaw	0.0438546
RElbowRoll	1.18024
RWristYaw	0.606979
RHand	0

RIGHT HAND-HIGH-PLACE Movements and Parameters

Joint name	Angle(radians)
RShoulderPitch	-1.5708
RShoulderRoll	0
RElbowYaw	1.5708
RElbowRoll	0.0349066
RWristYaw	0
RHand	0

Each of these joints are given a start time due to which the movement syncs up correctly and the language has a time parameter that adds the feature of executing the motion with variable speed and wait time.

All these parameters and times are stored in arrays and send as arguments to the NAO robot which simulates these movements.

C. Language Interpretation

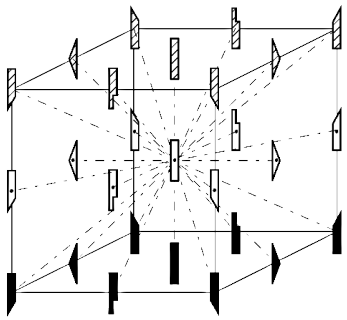
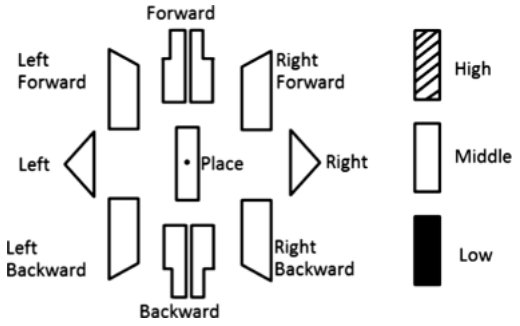
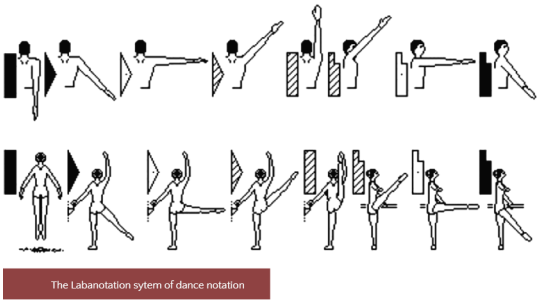


Fig. 2. Schematic representation of the Labanotaion points. [4]

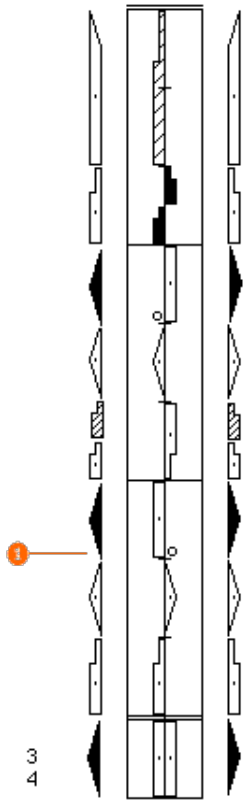
The way of representing the labanotation is in a table whose vertical axis represent the time (from bottom to top) and the columns represent the moves performed simultaneously. From the center to the outside the movements concern the body, the legs then the arms, then the hand, then the head.



The image above describes all the movements in an hierarchy and these 9 basic movements at each level is what we have used in our language subset to obtain the result which closely follows in with labanotation.

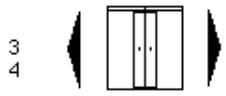


D. Language Mappings and Time dependence



The image above is how a typical labanotaion is given like, it is read from bottom to top.

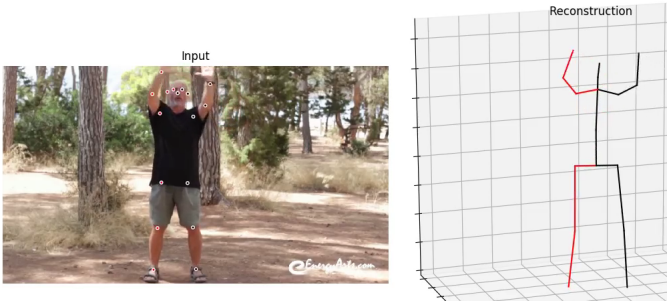
The starting position is given by-



Start with the feet together, knees straight. The arms are out to the sides, slanted down at a 45 degree angle.
Our language equivalent for starting position will be-
left hand, right hand,left leg , right leg. i.e.
" low L low R " is the user input and the output is -

E. Augmentation with Video

As of now we are using only hand movements for the system. The video processing gives us 2d co-ordinates which we pass to 3d reconstruction system and finally get a set of 3d points of each of the points marked the figure below.



The estimated 3d key points are used in synthesising formal dance notations by the process mentioned below: We take only the shoulder, elbow and wrist joints and store them in an array. The mapping is done by making 2 vectors out of these 3 points then checking which part of the labanotation cube it fits and hence generate a dance estimation for that motion in our interpretation of labanotation which in turn becomes the input to the previous system giving the final output.

F. Additional Features

1. The Language has an feature of looping added to it, which reduces the length of our interpreted labanotaion file

and increases readability in case we want to tweek the file manually for slightly different results, which will be shown instantaneously.
2. The whole code system is transported to Python and there is no use of any bash script as compared to its earlier version which has increased the speed of this whole process and is now more easily integrable with other python code for writing the code so that we can import all the video processing libraries in the same code, although for now we have kept the two processes in separate files for it to debug easily and possibly for keeping it open to future changes.
3. The motion can be done as slow as we want it to happen and the motion can also be asymmetric so it covers to whole sphere of movement and could be improved by adding more and more data points in that sphere.

IV. Limitations

1. Latency - As of now the whole process takes about a minute to generate a labanotation file which it turn runs through another python code , this limits the process to be done in real time but this could be improved by making some specific changes to the libraries we are using for the video processing part and make it faster.
2. Extension to the entire body is going to be a big goal for us, the idea is going to be the same i.e. to map the points first by the video processing and then take those points pass it through the python script to give us the appropriate labanotation which another python script will simulate on the NAO robot.
3. Balancing the Robot on all the motions is also a big challenge as currently there are some motions that challenge the robots ability to keep on standing and thus it requires a constant oversight from us to prevent it from falling.
3. The whole process runs essentially by 2 python files:
a)Video processing
b)Labanotation to movement
this could be reduced to make the whole process happend in one go and have an assurance that everything is in sync to each element in the two files at each update.

V. Future Goals

1.Generalising this to increase the number of points on the movement sphere.
2.Decreasing the Latency and try to make the process work in real time.
3.Balancing the center of gravity so that the robots never tips over and falls.

References

[1] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He, "Detectron." https://github.com/facebookresearch/detectron, 2018.
[2] D. Pavllo, C. Feichtenhofer, D. Grangier, and M. Auli, "3d human pose estimation in video with temporal convolutions and semi-supervised training," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
[3] R. A. Güler, N. Neverova, and I. Kokkinos, "Densepose: Dense human pose estimation in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7297–7306, 2018.
[4] "Labanotation." https://www.dcode.fr/labanotation, 2016.