Human Body Pose generation using Factor Graphs

Vikramjit Sidhu(2571661) and Garvita Tiwari(2571691)

Abstract

Obtaining 3D human body model data by scanning large number of people is an expensive task to do. We devise a method for generating such data, given some training data. We calculate the statistics of each body part. We use a factor graphical model to connect the body parts with Factor Graphical Model.

1 Introduction

We represent the human body using a Factor Graphical Model. Each node in the Factor Graph represents a body part, the body part is described using some parameters. We define our factor graph in the usual way using unary potentials for the nodes and pairwise potentials for restricting the values between the nodes. The training data is used to fit some distribution (Gaussian or mixture of Gaussian), this is then used to define the unary and pairwise potentials of the factor graph given in the figure 2.

We use Gibbs sampling to infer new poses once we learn the pairwise and unary potentials. An advantage of using a local statistical based model to represent the human body is that we get a larger variety of new poses than with a holistic body model. We study the behavior of our model for two types of distribution, a multivariate Gaussian and a Mixture of Multivariate Gaussians and by varying number of iterations in sampling process.

2 Project Description

2.1 Dataset and Parameters

The SMPL module was used as a framework for our project. SMPL models the human body using parameters such as pose, joint locations, vertices of 3d mesh etc. Our usage of SMPL is varied. We use it to load the body model data, to obtain the faces and vertices which are required by the mayavi library to visualize the body model.

For our implementation, we use pose to parametrize each body part. Alternately, we could have used parameters such as joints locations, or a tapered cylinder like structure to model our body parts, as in [2]. However the pose parameter seemed sufficient to represent each part.The pose parameter for a body part is defined by joint rotation angles (r1,r2,r3). All poses are calculated relative to a 'rest pose' which is shown in figure 2. When the body model is in the rest pose, each body part will have pose values (0,0,0). The overall pose of the body model is described by a 72 X 1 array of rotation vectors. Three consecutive values belonging to one body part for a total 24 body parts[1].

The main issue initially was to figure out, which 3 values corresponds to which body part. For this, we initially changed the overall pose array to the rest pose, figure 2. We then changed turn by turn 3 values to (0,0,pi/2) in the pose array, and checked which part was changed. This process is shown in figure 1. From this we got which values in the pose array correspond to which body part, we could finally construct our Factor Graphical Model which is shown in figure 2.

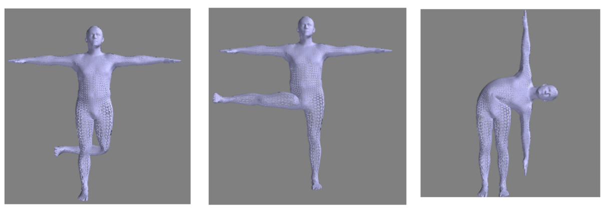

Figure 1: Pose of lower torso, left knee and right thigh is changed to (0,0,pi/2)

We generated the pose and joint data using the CMU dataset [5] and dyna files available to us. The pkl files in CMU dataset contained the pose values for various human body motions like running, dancing, jumping etc. taken from consecutive video frames. To get non redundant data, we skipped few frames(pose values) and stored every 10th or 20th pose data for a given human body movement.

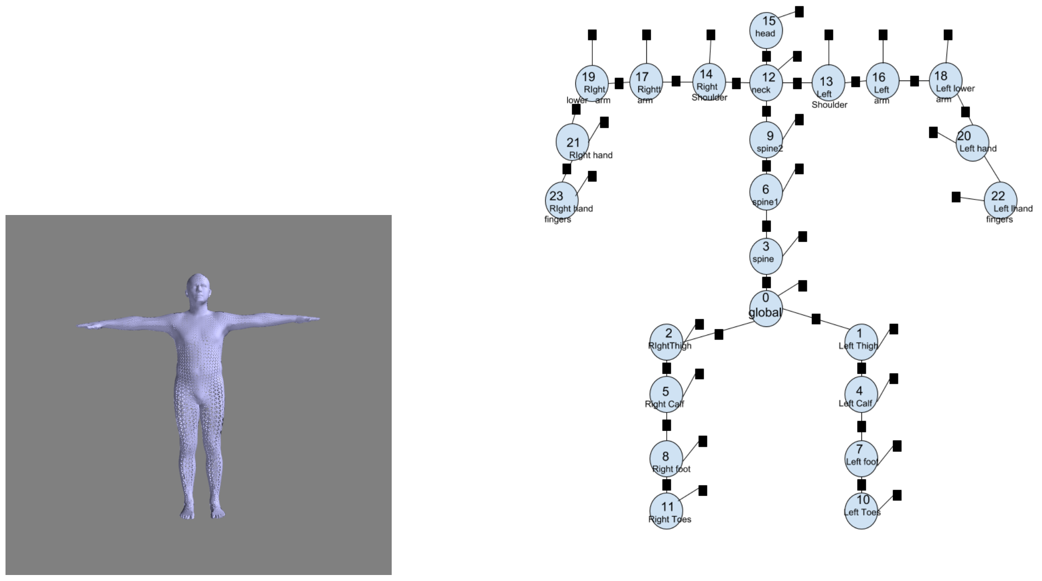

Figure 2: Rest pose of human body. At this pose all the pose parameters $[\theta_1\theta_2\theta_2] = [000]$ Right figure: factor graph representation of human body model

2.2 Constraints

We used kinematic constraints between body parts, this basically restricts the motion of body parts which are connected. These constraints are encode in our model by the pairwise potentials which are defined by Gaussian distribution(or mixture of Gaussian).

Figure 3 shows the histogram of joint rotation vector of few body parts. It can be seen that each rotation vector can be approximated by either a Gaussian or mixture of Gaussian distributions.

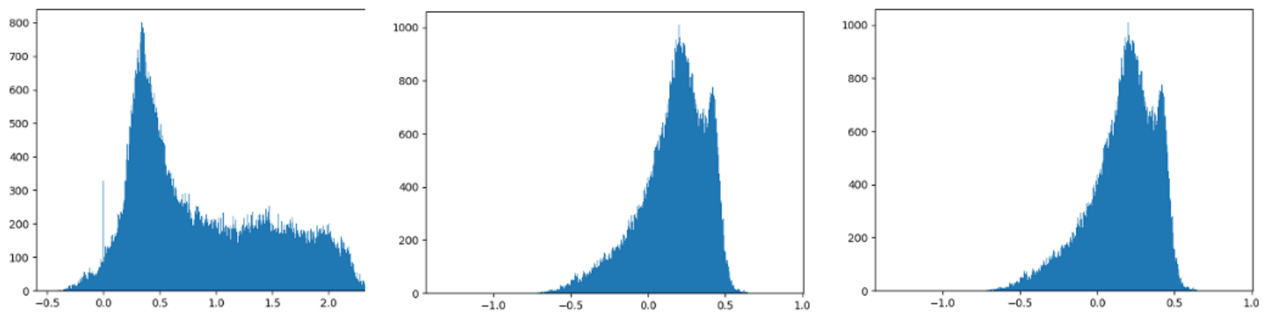

Figure 3: Histogram of one rotation vector of left upper arm, spine and Right shoulder respectively

2.3 Learning Kinematic Priors and Conditionals

The graphical model represented by factor graph in figure 2 has two type of factors, unary potentials $\phi(X_i)$ and pairwise potentials $\psi(X_i, X_j)$, where X_i X_j are the state variable defining node i and j respectively and node j is the neighbor of node i[2].

$$X_i = [\theta_1\theta_2\theta_3] \tag{1}$$

$$\phi(X_i) \sim \mathcal{N}(\mu, \Lambda)\,. \tag{2}$$

$$(\psi(X_i, X_j)) \quad \sim \quad \mathcal{N}\left[\left(\begin{array}{c} \mu_{im} \\ \mu_{jm} \end{array}\right), \left(\begin{array}{cc} \Lambda_{ii} & \Lambda_{ij} \\ \Lambda_{ji} & \Lambda_{jj} \end{array}\right)\right]$$

In equation 2, Λ_{ii} and Λ_{jj} are covariance matrices of parameters of node i and j respectively, Λ_{ij} and Λ_{ji} are covariance matrix of parameters of node i relative to parameters of node j and vice-versa respectively. In our approach we are using only pose parameters right now, so the dimension of covariance matrix in unary is

3 X 3 and in pairwise potential is 6 X 6. this approach can be extended to using more parameters like joint location, etc..

After successfully implementing inference using a Gaussian distribution, we decided to try an alternate approach. The histogram in figure 3 reflects that local pose parameters could be better approximated by a mixture of Gaussians. We changed our pairwise and unary potentials to have a more general form, i.e. a mixture of Gaussians. We initially choose a mixture of 3 Gaussians. However we also compared the results with a larger number of Gaussian mixture components.

The unary and pairwise potentials are modified as:

$$\phi(X_i) \sim \sum_{m=1}^{3} \delta_{im} \mathcal{N}(\mu_m, \Lambda_m)\,. \tag{3}$$

$$\big(\psi(X_i, X_j)\big) \quad \sim \quad \sum_{m=1}^{3*3} \delta_{ijm} \mathcal{N} \left[\begin{pmatrix} \mu_{im} \\ \mu_{jm} \end{pmatrix}, \begin{pmatrix} \Lambda_{iim} & \Lambda_{ijm} \\ \Lambda_{jim} & \Lambda_{jjm} \end{pmatrix} \right]$$

In equation 3, δ_{im} is the weight associated with each Gaussian curve in the mixture. To fit the data to a mixture of Gaussians, we used the algorithm of Expectation-maximization. This algorithm is implemented in scikit-learn[4] and it gives mean, covariances and weight of each Gaussian Component.

2.4 Sampling or Approximate Inference

After learning the conditionals and priors, we then performed approximate inference on the graphical model to sample human body poses. This was performed using a modified version of Gibbs sampling over the factor graph figure 2.

2.4.1 Sampling from Gaussian Distribution

Algorithm 1 underlines the process of obtaining a sample, assuming Gaussian distributions for the pairwise and unary potentials.

In the algorithm we initially generate a sample over all the nodes. We then run the sampling algorithm for some set number of iterations. In each iteration we select a node whose pose value is updated. The value of this node is updated using its unary potential and the pairwise potentials of that node with all its neighbors. The pairwise potentials are converted from Gaussian Distributions over 6 variables to a Gaussian distribution over 3 variables by conditioning on the neighbor of the node to be updated. The value of the neighbor is obtained from the pose matrix. We then multiply the unary potential distribution and the conditioned pairwise distributions. A pose vector is sampled from this combined distribution which is the value of the node in this iteration.

However, the implementation of the algorithm was limited by the fact, that Λ_{22} might be a singular matrix. Hence calculating the matrix inverse was not possible. Two possible approach to overcome this were either to perform PCA and find new components with maximal variance or to calculate the pseudo-inverse instead of the inverse. Using the PCA approach is computationally expensive as it requires finding principal components and then re-projecting to get samples and visualize the SMPL model. In this approach even the re-projection step could have introduced unavoidable error. Also there are very few cases, where the covariance matrix comes out to be singular (like near torso, fingers, etc.) Hence it was better suited to use pseudo-inverse method in our case.

2.4.2 Sampling from Mixture of Gaussian Distribution

While sampling from mixture of Gaussian, we first have to select the Gaussian mixture component and then sample from it. The method for selecting the Gaussian mixture component is given in algorithm 2. Sampling from a mixture is more computationally expensive from the Gaussian approach. This is because when multiplying two distributions described by Gaussian Mixture components we have to find the combination of all the components. For example when using a Gaussian mixture with 3 components, for a node with two neighbors multiplying the unary potentials with the conditioned pairwise potentials, we get a Gaussian mixture with 27 components to sample from, i.e. sample from $\prod_{neighbors} p(X_m|X_{neighbors})$. This approach is more flexible, but computationally complex, especially in case of large neighbors at spine/lower torso(3 neighbors) and neck(4 neighbors). Everything else is same as Algorithm 1 for Gaussian distribution.

Algorithm 1 Sample Human pose from Local Statistical Model(Gaussian Distribution)

```
Require: all unary and pairwise potentials
  initialization: state of all random variable $sampled - data \leftarrow 0$
  while $iterations \leq 10000$ do
    Randomly select $m \quad \forall \quad m \in 0, 1, 2, ...., 23$
    Distribution of $X_m \sim \mathcal{N}(\mu_m, \Lambda_m)$
    clear all $\mu$ & $\Lambda$
    $\Lambda_m \leftarrow covaraince(m)$ $\mu_m \leftarrow mean(m)$
    $resulting\ distribtion,\ p_{resulting} \leftarrow \mathcal{N}(\mu_m, \Lambda m)$
    for all neighbor of $m$ do
      $data \leftarrow colomn - stack(m, neighbor)$
      $\mu \leftarrow [mean(data)]$
      $\Lambda \leftarrow [covariance(data)]$
      $\mu_1 \leftarrow \mu[0:3]$
      $\mu_2 \leftarrow \mu[3:end]$
      $\Lambda_{11} \leftarrow \Lambda[0:3, 0:3]$
      $\Lambda_{12} \leftarrow \Lambda[0:3, 3:end]$
      $\Lambda_{21} \leftarrow \Lambda[3:end, 0:3]$
      $\Lambda_{22} \leftarrow \Lambda[3:end, 3:end]$
      $a \leftarrow sampled - data[neighbor]$
      $\bar{\mu} = \mu_1 + \Lambda_{12} * \Lambda_{22}^{-1}(a - \mu_2)$
      $\bar{\Lambda} = \Lambda_{11} + \Lambda_{12} * \Lambda_{22}^{-1} * \Lambda 21$
      $p(X_m|X_{neighbor}) \sim \mathcal{N}(\bar{\mu}, \bar{\Lambda})$
      $p_{resulting} \leftarrow p_{resulting} \ \times p(X_m|X_{neighbor}))$
    end for
    sample from Gaussian distribution given by $p_{resulting} \sim \prod_{neighbors} p(X_m|X_{neighbors})$
    update $sampled - data[m] \leftarrow sample$
  end while
```

Algorithm 2 Sampling in mixture of Gaussian

```
Require: weights $\delta_i$ for each Gaussian component
  Generate a random variable $U \ \sim \ Uniform(0, 1)$
  If $U \in [\sum_{i=1}^{k} p_k, \sum_{i=1}^{k+1} p_{k+1}]$ interval, where $p_k$ correspond to the the probability of the $k^{th}$ component of the
  mixture model, then generate from the distribution of the $k^{th}$ component
```

3 Result and Conclusion

We used a dataset with 194388 poses[5] with various motion of human body and many repetitive poses. Two types of distributions were used to fit the training data, namely Gaussian and a mixture of Gaussians. We had hypothesized that the mixture of Gaussians approach would give us a wider variety of poses. However this was not true in practice, the multivariate Gaussian gave better poses.

3.1 Using Global statistics and mixture of Gaussian distribution

As a baseline for comparison we simply tried to find the statistics of how the poses are distributed. We approximated the pose parameters using a mixture of Gaussians over 72 variables. We then sampled from this distribution to obtain the pose matrix. Samples from this method are shown in figure 4

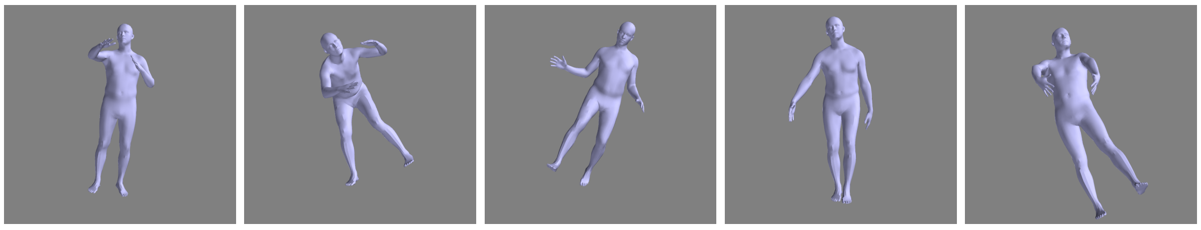

Figure 4: Sampling from global multivariate random variable distribution

3.2 Using Gaussian

It was observed that when using the Gaussian distribution to fit the data, we get very realistic and unique poses. Moreover the algorithm is computationally very fast. Row 1 in figure 5 shows the result for 12000 iterations and row 2 shows the result for 24000 iterations. This method is fast and generates realistic body poses.

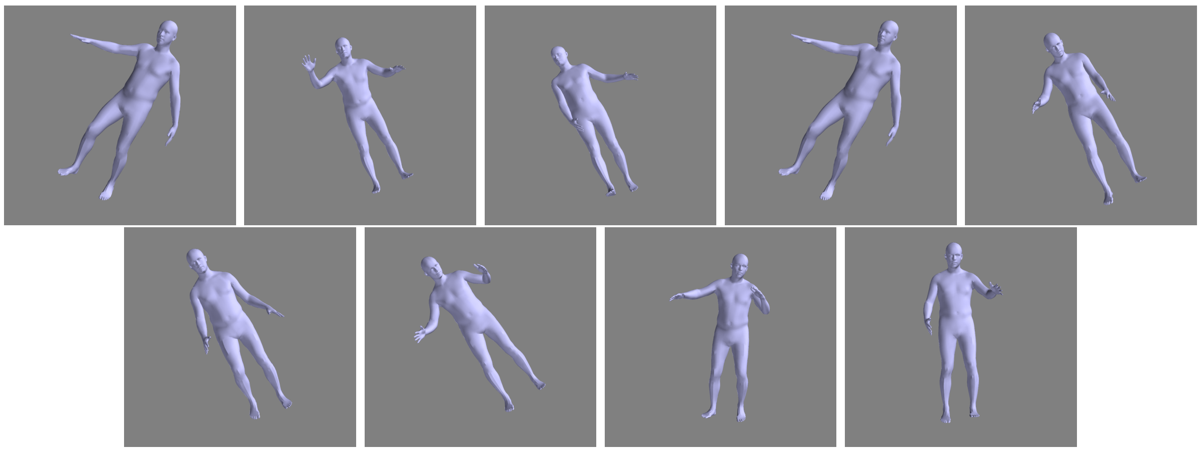

Figure 5: Samples generated form Gaussian distribution of variables in factor graph. Number of iterations in row 1 is 12000 and in row 2 is 24000

3.3 Using mixture of Gaussian

Although, sampling from Gaussian distribution gave good results, but we had thought that the data is better approximated by a mixture of Gaussians. The results are not satisfactory in the case of mixture of Gaussians. It was observed that by increasing the number of iterations in Gibbs sampling, the result got increasingly worse. For a small number of iterations, i.e. < 2000 we get reasonable human body pose, as seen in first human body in figure 6. However as we increase the number of iterations, results get worse. We have a few hypothesis as to why this happens:

- There are no global restrictions on the body model which enforces a constraint on all the body parts
- Overfitting of the data by increasing the model capacity in terms of the number of Gaussian components in mixture of Gaussians
- An incomplete understanding of how conditioning happens in a mixture of Gaussian components

However we did not have the opportunity to confirm any of these hypothesis.

Human models in first row of figure 6 corresponds to mixture of 6 Gaussian and second row corresponds to mixture of 3 Gaussian components. In theory, more number of Gaussian components gives better approximation of any data. In our case also we saw that a mixture of 6 Gaussian performs slightly better than a mixture of 3 components.

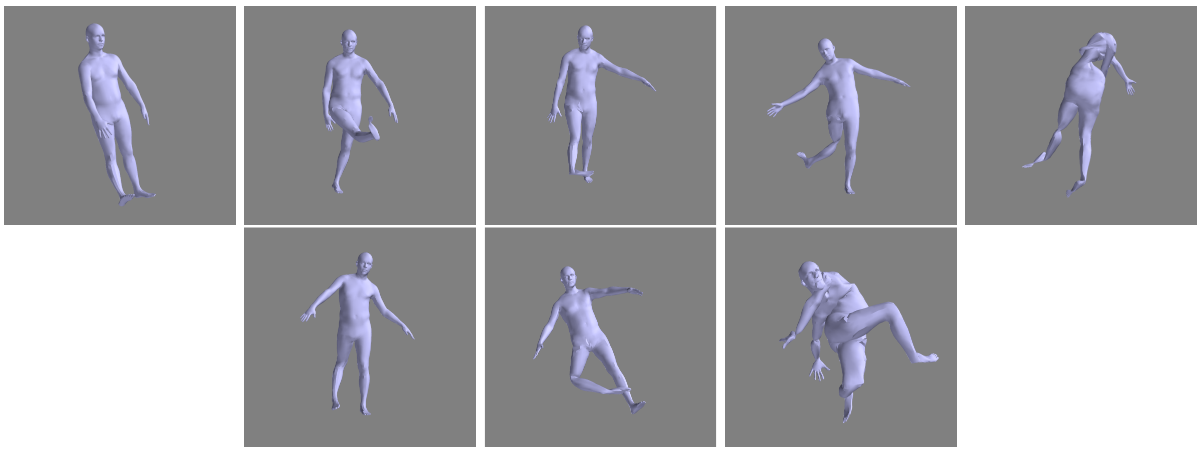

Figure 6: Samples generated form mixture of Gaussian distribution of variables in factor graph.

It can be seen from figure 4 , 5 and 6, that local statistics based sampling method is able to generate more new poses than the global one. This is because of the reason that Compared to Gaussian distribution approach, this approach is slower in sampling and training both, because of introduction of Gaussian components and exponential increase in Gaussian components after multiplying neighboring distributions.

4 Future Work

- Analysing the behaviour of Mixture of Gaussian, when it fails for large number of iterations.
- Some other parameters can also be used to define statistics of body parts, like tampered cylinder model to fit each body part.
- For a even more accurate modeling of human body, we can use some other constraints like inter-penetration constraints, which will introduce a loopy structure[2]. This approach has not been implemented yet, because our approximate inference currently works on tree or non loopy structure only.
- Some global constraint can also be added to model, to avoid haphazard results
- This model can also be extended for creating a dataset of 3D Human body motion

5 Running the Code

- To change the number of iteration over which we run the sampling over, change line number 7 in *perform_inference.py*
- To run the training and inference for a Gaussian distribution run the command *python main_fn_pose_only.py*
- To run the training algorithm for a mixture of Gaussians, run the command; *python perform_training_mix_gaussian*
- To change the number of Gaussian Mixture components, change Line 5 in the file *find_unary_potentials.py* and *find_pairwise_potentials.py*
- To perform the inference for a mixture of Gaussians, run the command *python perform_inference_mix_gaussians.py*

References

[1] SMPL: A Skinned Multi-Person Linear Model,
`https://ps.is.tuebingen.mpg.de/publications/smpl-2015`

[2] Leonid Sigal · Michael Isard · Horst Haussecker · Michael J. Black. *Loose-limbed People: Estimating 3D Human Pose and Motion Using Non-parametric Belief Propagation.*

[3] Wikipedia: Multivariate Normal Distribution,
`http://www-cs-faculty.stanford.edu/~uno/abcde.html`

[4] scikit-learn: Gaussian Mixture Models
`http://scikit-learn.org/stable/modules/mixture.html`

[5] CMU Dataset
http://mocap.cs.cmu.edu/