CS 496 BTP II
Report

Hand Shape Estimation from Depth Images

Submitted in partial fulfillment of
the requirements for the award of the degree of

Bachelor of Technology
in
Computer Science and Engineering

Submitted by

Manas Bhargava
150050057

Under the guidance of
Prof. Parag Chaudhuri

Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Mumbai, Maharashtra, India – 400076
Spring Semester 2018-19

Acknowledgments

I would like to thank my supervisor, Prof. Parag Chaudhuri, for the patient guidance, encouragement and advice he has provided throughout this project work.

I would also like to thank Pratik Kalshetti for helping me throughout this project work . He has always been my first point of contact throughout my project and has helped me a lot in clarifying any conceptual doubts and answering my queries.

Lastly, I would like to thank my family and friends for their constant support.

Manas Bhargava
Senior Undergraduate Student
Department of Computer Science and Engineering
IIT Bombay
2^{nd} May, 2019

Abstract

This report discusses the work done on estimation of hand shape parameters from depth images, reconstruction of the shaped hand mesh and articulation of hand mesh based on the pose of the hand in the current frame. It builds upon the previous work that I have done in my BTP-I project. As a whole we worked on the problem of hand pose estimation. In which we are given a set of user's hand performance as captured by a Kinect camera in the form of depth images. Thus, replication of actual user's hand action include accurate estimation of shape of hand by a high quality mesh along with accurate estimation of each of the joint position on every frame and then animating the mesh to artificially resemble the action performed by the user. To achieve this task we have divided this problem into three separate parts. The first part deals with estimation of shaped hand mesh which is personalized for every user. The second task is to predict all the joint positions of the hand for every frame that represents different motion of users hand in front of camera. The third task is to take the predicted joint positions from every frame and apply it to the obtained shaped hand mesh and articulate it accordingly and thus generating the posed hand mesh.

This report discusses the the portion of the work done by me which includes estimation of the shape of the hand's mesh model, generating the mesh model from it, finding the joint position from this hand model and then animating this mesh model given the pose of hand in that particular frame.

Contents

Chapter 1

Introduction

The hand is the primary operating tool for humans. Therefore, its location, orientation and articulation in space is vital for many potential applications in robotics such as object handover, learning from demonstration and indirect interaction of robotic arms from user hands. It is also used in sign language and gesture recognition and in recent years it has also found its usage in Virtual Reality (VR), Augmented Reality (AR) and Human-Computer Interaction where an artificial model of hand controlled by the user hand is used to interact with the other objects in the artificial world.

In the past few years major progresses have been made in pose estimation thanks to the emergence of depth sensors and deep learning models. These techniques have been successfully used in designing morphable models for the human body. Starting from the face models of Blanz and Vetter [1], and proceeding to combined shape and pose models of the full body [2, 3, 4, 5] , such models are now starting to see commercial deployment for applications including virtual shopping (e.g. Metail), performance capture (e.g. faceshift), and video gaming.

The hand is in some senses ideal for such modeling since it is normally unclothed and thus has huge potential for natural 3D user interfaces. Ballan et al. [4] demonstrated that extremely robust hand tracking is possible given a user-specialized hand model, but acquiring the model requires manual rigging and a multi-camera capture setup. Taylor et al. [6] demonstrated acquisition of a user-specialized model from a single depth camera, but require long calibration sequences in which all degrees of freedom of the hand have to be exercised.

Recent developments in hand motion capture technology have brought us a step closer to achieving effective tracking, where hardware solutions such as data-gloves, reflective markers and multi-camera setups have been shelved due to their invasiveness and cumbersome setup. Hence, a single camera has become the standard acquisition device, where depth cameras have taken a solid lead over color imagery because of the their superiority in representing the position of the object more properly than their latter counterpart [7].

In the recent years people have worked with two common approaches to solve the hand tracking problem from depth images: the model-based generative tracking and the discriminative based tracking. Both approaches have their drawbacks: for the first case it is difficult to use complex hand models as generative models since it is difficult to optimize the constraints over positions from highly complex large dimensional model and thus simple model representation of hands are used [8]. Thus high quality reconstruction of hand mesh is not possible. On the other hand the latter faces difficulty in addressing the case of self occlusions , lower resolution, clutter and object surface interaction where

it is difficult to resolve the dis-ambiguity of joint positions from a single depth image [7]. Also these techniques just tracks the positions of joints and thus mesh cannot be reconstructed from such approaches.

Modern techniques combine the benefits of both generative tracking and discriminative detection into a unified framework to yield better results (e.g. Taylor et al. [9]) utilized discriminative techniques (e.g. Valentin et al. [10]) to identify a coarse pose, followed by a generative stage (e.g. Tkach et al. [11]) to refine the alignment and obtain a precise pose estimate. But still the hand model used by them is primitive based on simple geometric models thus incapable of producing high quality meshes.

Using the above methods as our motivation, we have tackled this task by dividing it into two individual problem of learning hand's shape and its joint positions separately and then combining in a unified network with the hope that tackling the two problems separately and efficiently would help us to get better results in both of them and then by combining them together we would be able to have a better and robust system to solve this overall problem of hand pose estimation.

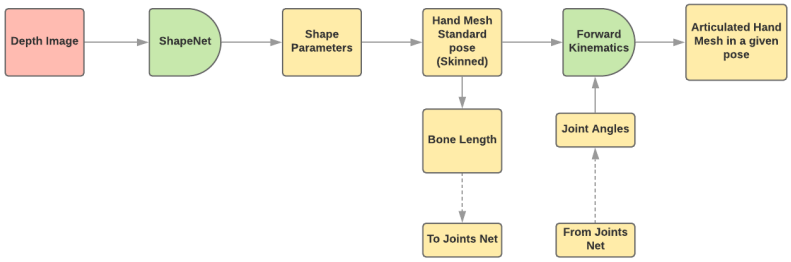


Figure 1.1: **Portion of the pipeline that I have worked on. Refer to 6 for overall pipeline of our work.**

Chapter 2

Related Work

The task of hand tracking is primarily done using two methods - Generative methods and Discriminative methods. Generative methods primarily depends on first forming a parametric hand model and then inferring the joint positions from them. While discriminative methods just rely on predicting the joint positions directly from depth images. In this section we will explain these methods which are predominantly expressed under the topic of hand pose estimation and then discuss the related work done for hand shape estimation.

2.1 Hand pose estimation :

Hand pose estimation methods can be primarily classified with respect to the input as depth, monocular RGB, multi-view, and video-based. Given the low cost of depth sensors and its benefits against normal color images, there has been a vast amount of work on hand pose estimation based on depth images, and as seen above this can be further classified as being either generative (model-based), discriminative (appearance based), or both (hybrid) [12] . An additional classification can be made based on how the input is mapped to the output : 2D-to-3D lifting similar to Bogo's work. [13] or direct 3D mapping based methods [14].

Generative Approaches: Melax et al. [15] formulated the hand optimization as a constrained rigid body problem. Schroder et al. [16] suggested to optimize this constrained rigid body problem in a reduced parameter space and Tagliasacchi et al. [17] combined ICP with temporal, collision, silhouette, kinematic and data-driven terms to track with high robustness and accuracy from a depth video. Sharp et al. [18] enhanced this approach with a smooth model and the possibility of re-initialization. Particle swarm optimization (PSO) approaches have also been used, requiring extensive rendering of an explicit hand model in various poses [19], estimating ground truth [20] for the NYU dataset [21], or combining it with ICP [22] to increase its robustness. Taylor et al. [23] have used minimization of an error between a realistically synthesized and real depth image as a formulation of their optimization problem.

Discriminative Approaches: Oberweger et al. [14] showed how to boost the prediction performance by a projection to a reduced subspace before the final regression, through a bottleneck layer. Zhou et al.[24] predict joint rotation angles (similar to our work) by proposing a forward kinematic layer, coupled with a physical loss to penalize angles outside a specified range. Similarly, Dibra et al. [25] map the position to angles and show how to refine their CNN on unlabeled depth input images. Apart from CNN-s,

there also exist methods that utilize decision forests to make a 3D pose prediction [26].

Hybrid Approaches: Sometimes, CNN predictions are complemented with an optimization step. Tompson et al. [21] first predict hand keypoints and optimize for the actual pose using inverse kinematics. Mueller et al. [27] fit the hand skeleton to 2D and 3D joint predictions from a CNN. Ye et al. [28] combine CNN-s and PSO in a cascaded and hierarchical manner. Sinha et al. [29] first reduce the dimensionality of the depth input through a CNN and then adopt a matrix completion approach with temporal information to optimize for the final pose. Oberweger et al. [14] use a deep generative neural network to synthesize depth images, which are utilized to iteratively correct a pose predicted by another network during testing.

Video-Based Methods: Since RGBD sensors are not always available, further methods have been proposed, that utilize RGB images in combination with temporal information. La Gorce et al. [30] use texture, position and pose information from the previous frame to predict the current pose. Romero et al. [31] exploit temporal knowledge to guide a nearest-neighbor search.But all these methods face difficulty in solving the problem of obtaining a first estimate.

Multi-View-Based Methods: Another approach involves the use of multiple cameras to compensate for the lack of depth data, alleviating the problems with occluded parts. Zhang et al. [32] utilized stereo matching for hand tracking, Simon et al. [33] apply multi-view bootstrapping for keypoint detection, and Sridhar et al. [34] estimated 3D hand pose from multiple RGB cameras, with a hand shape representation based on a sum of Anisotropic Gaussians, whereas in this work [35] he combined RGB and Depth data to obtain a richer input space.

Image-Based Methods: Due to the larger availability of regular color cameras, as opposed to the above-mentioned methods, One of the first single frame based hand detection works, from Athitsos and Sclaroff [36] utilize edge maps and Chamfer matching. Zimmermann et al. [37] utilized CNN-s and synthetic datasets for 3D hand pose estimation from Monocular RGB image. They split the prediction into a 2D joint localization step followed by a 3D up-lifting, and use their own synthetic dataset to complement the scarcity of existing datasets. Methods based on Variational Autoencoders [38] for cross-modal learning and GANs [27] for learning a mapping from synthetic to real hands data, are also used to tackle the same problem of 3D hand pose estimation from RGB images.

2.2 Hand shape estimation methods :

Learning a lower dimensional parametrization of shape from examples of the range scans or other 3D data has proved effective in creating generic morphable models for human bodies and faces. Having built such morphable models, impressive applications e.g. for fitting body shape to a monocular depth sequences or more precise body or face tracking have been demonstrated.However, despite a long history of successes for faces [1] and bodies [2, 3, 4, 5], there are very few existing statistical shape models for hands. This suggests that this is a challenging problem, where existing techniques for whole bodies do not directly transfer. Previous works in this topic by Allen's [39], Taylor's [23] and Khamis work [40] are the notable contribution in the field of hand shape meshes extraction from depth images.

Allen's work : represent the model as an adaptation of a standard subdivision surface

model with linear blend skinning. Crucially, however, their adaptations are displacement maps on top of a base surface. The displacements must be limited in magnitude to avoid self-intersections, and their shape basis is forced to coincide with the input scans. Also their optimization step was chosen such that it converges to a local optimum instead of a global optimum.

Taylor's work : - A more automated technique was presented by Taylor et al. which generates personalized hand models given noisy input depth sequences where the users hand rotates 180 degrees. whilst articulating fingers. A continuous optimization that jointly solves for correspondences and model parameters across a smooth subdivision surface with as rigid as possible (ARAP) regularization leads to high-quality user-specific rigged hand models, though not a shape basis. Whilst the process is automatic, the hands are required to cover the full range of articulations, and longer sequences are required, leading to more complex capture requirements and more costly optimization.

khamis work - Their work is based on a skeleton-driven morphable model that is learned from sparse and noisy data. Although their model is very efficient at test time, being linear in the number of basis components and requiring only a few components to accurately describe a wide variety of human hands. But like the other models it has a very coarse resolution. Thus there is a need for a model which represents a high resolution mesh model and can be obtained at real time from a depth image.

In this report we present our model that gives the output mesh for given input depth image in real time rather than giving the hand shape model offline. That is once our model is trained, it predicts hand shape parameters from which we generate the hand shape model on the fly as soon as user shows his hand.

Since we have complete knowledge of the hand model we can generate a skeleton model from the mesh and skin it with our mesh model. We also obtain the joints angles for every depth frames that is shown to us by a separate pose estimation model (refer to appendix chapter-6). We use these pose values to animate the hand model to match the action as performed by the user. In the next section we will discuss the procedure behind our hand shape estimation module and animation module.

Chapter 3

Hand Shape Estimation

In this chapter we will discuss the pipeline of Hand Shape estimation. We will first define our parametric hand model and compare it with our previous parametrization approach. We will then discuss the procedure of generating dataset based on this parameterization model. After that we will discuss the Network architecture that we have used for predicting the shape parameters given the depth images of a user. Later we will see how these networks are trained. We will end our discussion on Shape estimation by discussing the articulation pipeline and prediction of joint position given its pose.

3.1 Problem with previous parameterisation model of Hand Shape

In my previous work (BTP-I), We have parameterized the hand model using pca based method. Where we created a dataset of 1400 different hands of varied shapes in standard pose, represented hands as vectors and then used these vectors to learn a PCA based representation of hands using 30 PCA vectors with different weights. Thus, each of these hands were parameterized using these 30 vectors. After that we trained a CNN based network to predict these PCA weights given the depth image. Though this approach worked well for synthetically generated hands in standard pose, they failed to give good quality results on real people hands let alone for depth images in which hands were not in standard pose. Also this representation did not directly tell us about properties of a generated hand i.e. there was no physical meaning as to what would be the parameter values for a fat baby hands or old person hands who has big fingers.

3.2 Parameterisation of Hand Shape and obtaining the Shaped Mesh Model

Our new parametric model uses simple hand related properties to model the hand viz. **Overall hand scale, finger length scale, finger radial scale and uniform palm scale**. All these scales are normalized so that 0 represents smallest possible scale while 1 represent biggest possible scale for human hands. We also have a standard hand mesh model for which we have complete knowledge of the face structure. Thus giving us the complete knowledge of which vertex belongs to which part of the finger. Now given a set of shape parameters, we first denormalize them to obtain the relative scales relative to

our standard mesh model. Overall scale is applied to scale the overall hand mesh. After that for each finger we apply finger linear scale and finger radial scale. For this we first obtain the connecting point which joins the points of that finger to the palm. This point and the topmost point in the finger form the finger's linear axis. The radial finger axis is thus any axis perpendicular to the finger's linear axis. The finger length scale is applied along the linear axis scaling the finger along that direction and the finger radial scale is applied along the radial axis thus scaling the finger along the radial axis. The palm scale is applied to all the points that are part of the palm with scaling done with respect to the centroid these points. Thus after all these scaling operations are done the standard hand mesh gets converted to scaled hand mesh.

This parameterisation helped us in solving the problem of meaning-full parametrization of hand. Thus now depending upon the hand model, whether it be fat baby's hand or man's hand with long finger, the user can specify the corresponding parameter value to obtain the hand model.

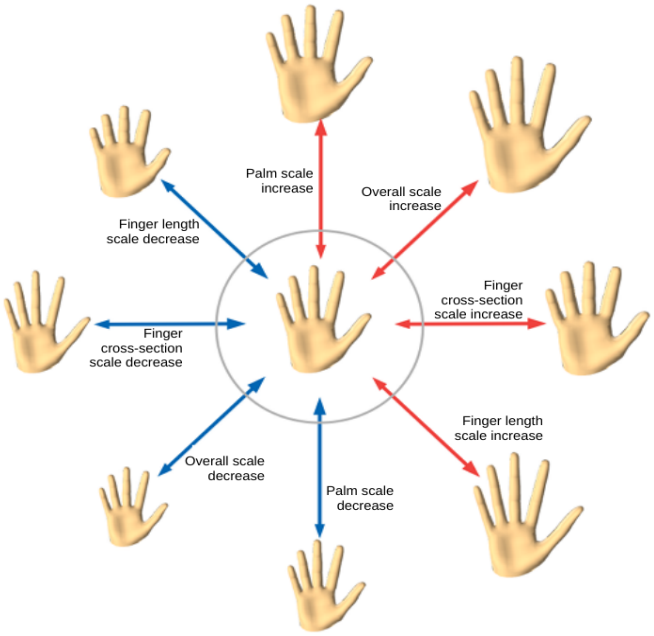


Figure 3.1: **Hands generated using different parameter values.**

3.3 Generating dataset

In this section we will discuss the different datasets that we generate using this parameterization. While solving the task of hand pose estimation we have used Bighand dataset [41], NYU dataset [21] and MSR dataset [42] which contains depth images of different people in varied poses. Along with the depth images we are also given the corresponding joint positions for that image. Our aim is to obtain the hand shape parameters that approximates the shape of the hand that is there in the depth image. To obtain these shape parameters we use **offline fitting** method that is discussed in the section 3.4. We then use these approximated shape parameters to reconstruct the actual mesh of the

hand for that shape. Using the inverse kinematics on joint positions we recover the pose of the hand. We then apply this pose (angle) using our forward kinematics model to transform the skinned mesh. Thus at the end of this pipeline we obtain the shaped and posed mesh which has the shape that represents the shape properties of hand and pose which represents the pose of the hand. From now on we will call this mesh as shaped posed mesh. Thus for each of these datasets we now also have the shape parameters along with the joints positions for the corresponding depth images. Apart from these datasets which we call the real dataset, we also construct a synthetic dataset that helps in improving the training of our shapeNet. We obtain the anthropomorphic constraints for the shape parameter space and pose parameter space which are then used to generate human hands of varied shapes and poses. We then randomly sample from this shape space and pose space to obtain 10^6 different posed shaped depth images along with corresponding shape parameters. This approach of having both posed and shaped depth images in our dataset is a direct upgrade over our previous approach where we just generated the hand models in standard pose, which rendered our previous parametrization model ineffective for depth images which were not in standard pose.

In the later sections we will see how we these estimated shape parameters are used to train shape estimation network i.e. ShapeNet.

3.4 Offline Shape fitting

In this section we will discuss the procedure that we use to estimate the shape parameters given the elements of dataset. We have depth image and their corresponding joint position to estimate the shape parameters. We want shape parameters such that the depth image that it generates after posing matches the depth image that we have given. Previous work done by Tkach et.al. [8], [11] and Taylor [23], [6] formulates this task of obtaining shape meshes from depth images as an energy minimization problem. But the hand model that they were dealing with were simple geometric models and most importantly the depth images that they have were properly masked. The datasets that we are working on namely NYU, BigHand and MSR does not have masked depth images thus rendering these techniques useless for our task.

Instead we make use of the joint positions for every depth image to estimate the corresponding shape parameters. We use offline fitting technique where we try to search these parameters by using **Hill Climbing** [43] based approach. We start off with shape parameters representing the mean human hand's shape, form the shaped mesh and then estimate the bone lengths from this shaped mesh. Since we know the structure of the hand i.e. which vertex represent which part of the hand, we can thus estimate the different joint positions of the hand in standard pose and thus calculate the bone lengths for the hand. Since the bone lengths are invariant to the pose of hand. We can thus compare the bone lengths obtained via our shaped mesh and the ground truth bone lengths that were part of the dataset. Thus, we formulate the reward function for a particular shape parameter as follows -

$$Reward : -\sum_{i \in 5} || \sum_{j \in 4} GBL(i,j) - PBL(i,j) ||$$

where i is the finger index, j in the bone index of that finger, GBL grund truth bone length and PBL is the predicted bone length.

We divide our Hill climbing procedure into two steps- Coarse parameter tuning and fine parameter tuning. During the coarse parameter tuning we just change the overall shape while defining the neighbourhood of the given shape parameter. While in fine parameter tuning we vary the other shape parameters to obtain the neighbourhood of the given shape parameter. For both of the steps we run the hill climbing algorithm [43] with the reward function and neighbourhood set as defined above. Thus at the termination of this algorithm we obtain the optimal shape parameter for that element of the dataset. Each element of this dataset takes around 0.5 seconds to complete.

3.5 ShapeNet

In this section we will discuss our CNN based model which is used for prediction of hand shapes from depth images. We will first discuss the network architecture and then training of the CNN model both individual and combined training with JointsNet.

3.5.1 Network architecture

To counter the difficulty of transfer learning between the synthetic dataset and real dataset, we modified our CNN network (ShapeNet) such that it utilizes both synthetic depth images and real depth images for training, retains the information from both the datasets and is able to generalize well on unseen real depth images. Thus our modified ShapeNet is made of three parts - The feature Extractor Network, Feature Mapper Network and the Estimation Network based on the work done by Mahdi Rad et.al. [44]

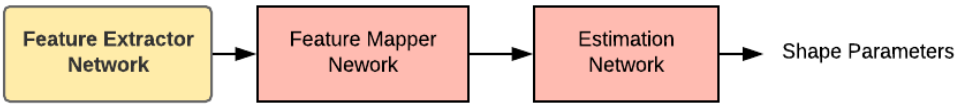


Figure 3.2: **ShapeNet**

Feature extractor network architecture - The Feature Extractor is used for training both real and synthetic dataset. It's task is to extract out convolution based features from the depth images. This network takes depth image of size 128 X 128 as input and outputs feature vector of size 512. The number of filters in the conv2D layer is doubled each time. Leaky ReLU is used for non-linearity at every layer.

The mapper Network - If real dataset is used then Feature Mapper Network converts the real extracted convolutional features from Feature Extractor network to synthetic convolutional features. Thus this feature mapper network maps real features of size 512 to synthetic features of the same size. It is composed of two residual blocks (as shown in the right image of whose outputs are added to get the final network output.

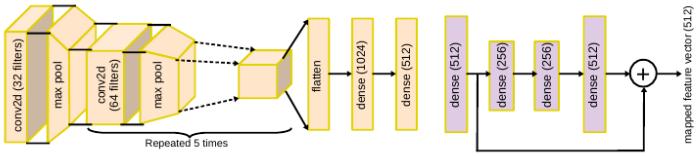


Figure 3.3: **Feature Extractor (left) and Feature Mapper Networks (Right)**

The estimation network - Synthetic convolutional features are passed to Estimation network which predicts the Shape Parameters. It is made up of two dense layers of size 256 and 4.

3.5.2 Training

ShapeNet is pre-trained with completely synthetic hand data, in which shaped posed depth images are generated by randomly sampling the space of valid shape space and pose space as discussed in Section 3.3. A synthetic depth image is constructed from the posed hand mesh. For this pre-training step only feature extractor network and Estimator networks are used. This pre-training improves the performance of ShapeNet and acts as a good initializer for training on real depth images. Then entire ShapeNet is trained end-to-end using the real dataset . During this training, each training input instance contains corresponding pairs of real and synthetic hand data. The real data in the input is used by the all 3 networks while the corresponding synthetic data is used only by the feature extractor network and Estimator network. For synthetic dataset we just use the MSE loss between the predicted shape parameter(PSP) and ground truth shape parameter(GSP). Thus the overall loss for training on the synthetic dataset is defined as

$$Loss = \sum_{i \in 4} ||PSP(i) - GSP(i)||^2$$

During training on real dataset we use MSE loss between predicted shape parameter from both real depth image (RSP) and synthetic depth image(SSP) to the ground truth shape parameter(GSP) obtained from offline shape fitting. Along with this we also have a MSE loss between the synthetic features (SF) from feature extractor network (for synthetic image) and synthetic features (RF) from the mapper network (for real image). Thus the overall loss for training on the real dataset is defined as -

$$Loss = \sum_{i \in 4} ||RSP(i) - GSP(i)||^2 + \sum_{i \in 4} ||SSP(i) - GSP(i)||^2 + \sum_{i \in features} ||SF(i) - RF(i)||^2$$

During training of ShapeNet, each epoch takes around 1 hour on nvidia-1080 GPU.

3.6 Post Processing Step

3.6.1 Animating the hand mesh model

Given the shaped hand mesh, we then articulate our mesh by using the pose information of the hand (prediction from Joints Net refer to Chapter 6 for further details). For this

we use the predefined skinning weights that were computed using blender and were later manually fixed to avoid any artifacts. We then articulate this shaped mesh with help of pose information and skinning weights using Linear Blend skinning. This entire part is implemented as a differentiable function in TensorFlow. (refer to Section 6.2 for further details.) Thus at the end of this step we obtain the posed shaped hand mesh.

3.6.2 Predicting the joint positions

The next task is to infer the joint positions and bone lengths from the posed shaped mesh. These joint positions and bone lengths play an important role in combined training of JointsNet and ShapeNet (See Section 6.1.2 for further details). These joint positions are to be such that they match the ground truth annotations that are there in the dataset. The main problem with fixing the indices of these joints positions is that all 3 datasets have different positioning of their annotations. Thus a single indexing of annotations does not work for all the dataset. Along with this there is an added complexity created by NYU dataset which does not have consistent location of annotations. Thus we decided to have different indexing for different datasets and manually found the indices (or group of indices) that best represent the location of annotation. Once we have the index information of joint position, we can use them to get the corresponding joint positions and hence the bone lengths.

Chapter 4

Results and Evaluation

4.1 Results

Once the network is trained we then use the ShapeNet to predict the shape parameters given a real depth image. All three subnetworks of ShapeNet are active during this inference task. By using this predicted shape parameters we can then generate the shaped hand mesh. We predict the shape parameters till 50 frames and after that we aggregate over the shape parameters that network has predicted so far and compute the final shape parameter which is used for remaining frames.

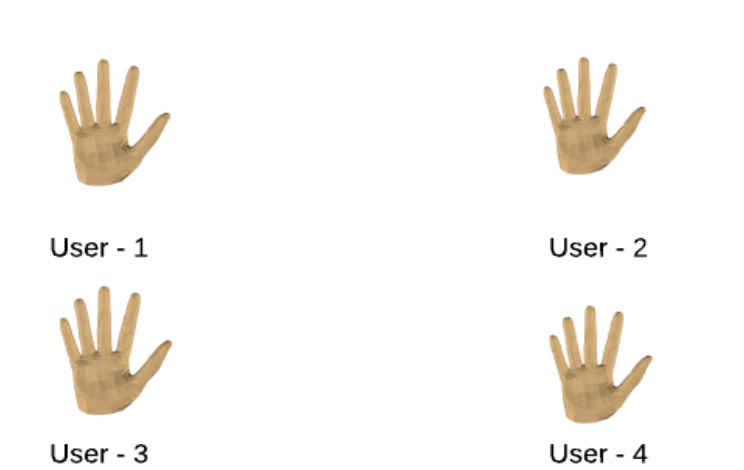


Figure 4.1: **Generated shaped hand meshes for different users**

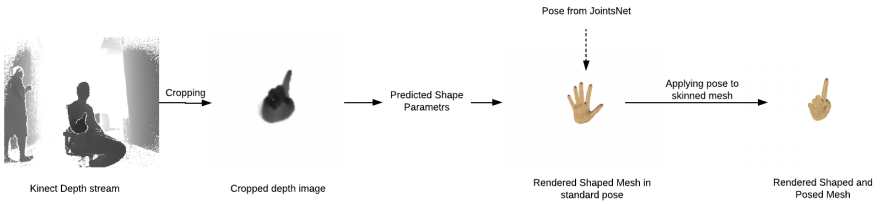


Figure 4.2: **ShapeNet Pipeline at run time**

4.2 Evaluating Shape Estimation

We have ground truth bone lengths for the BigHand dataset. We compute bone lengths for the various fingers using the Shapenet predictions from our model. We perform a comparison between the different versions of our model for mean bone length error. The fraction of bones for which the bone length is below a threshold is plotted in Figure 4.6. The average mean joint error in bone length measurement across all fingers for the BigHand test dataset is 2.81mm when combined training is done. Without the combined training, the ShapeNet model gives a mean bone length error of 3.75mm over the same data. We also compute an error, E_{m2d} to estimate how well the estimated shape fits the depth image. If D is the point cloud obtained from the depth image and M is a point cloud sampled from the predicted shape mesh then

$$E_{m2d} = \frac{1}{|M|} \sum_{m \in M} ||m - \Pi_D(m)||_2^1$$

where

$$\Pi_D(m) \colon d = \arg\min_{d_x} ||m - d_x||_2^1 \ \ \forall d_x \in D$$

E_{m2d} averaged over the entire BigHand test dataset is 11.75mm using combined model. The vertex to vertex loss between meshes generated by offline fitting using our shape space and the predicted hand shape meshes, for the same test data is 4.55mm using combined model. The same error computed by using ShapeNet alone is 8.46mm, thus establishing the clear benefit of combined training to shape estimation as well. (See Section 6.1.2 for more details about how combined training is done)

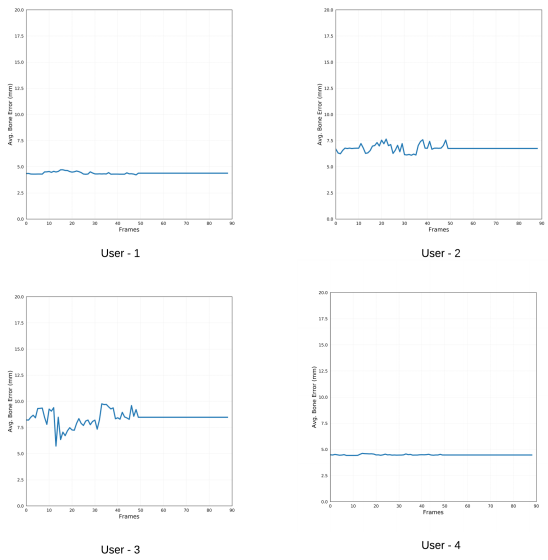


Figure 4.3: **Stabilization of Bone Length error over different frames for different users.**

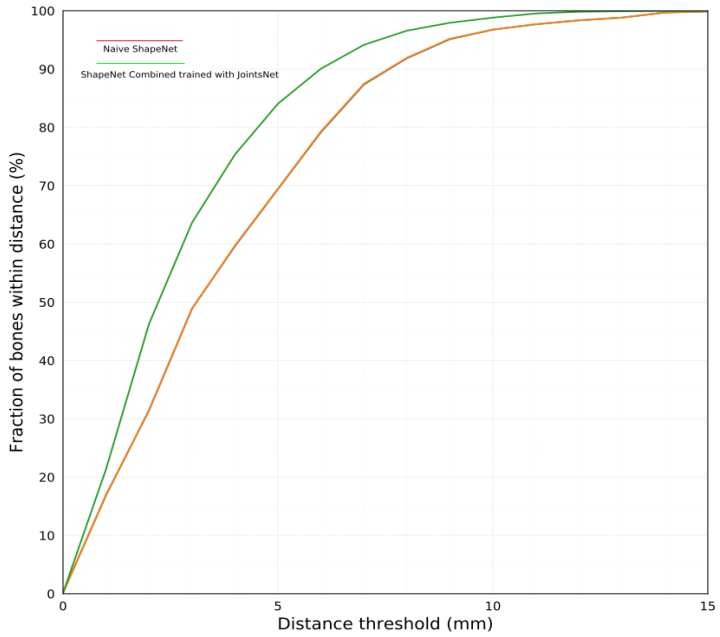


Figure 4.4: **Bone Length error when ShapeNet is trained alone and jointly trained.**

Figure 4.5: **Results for user - Manas.**

Figure 4.6: **Results for user - Pratik.**

Chapter 5

Conclusion and Future Work

To summarize our discussion, we have formulated the problem of hand shape estimation and animation of the entire model which forms the important sub-modules for the task of hand pose estimation. We have also presented the results that we have obtained on real human hand depth images for varied users. Based on our results we can conclude that our model works well on depth images of real human hands with varied shape and pose variations.

The offline fitting method that we have used just utilizes the joint position information and does not actually make use of depth images to approximate the shape parameters. Thus better approximates can be achieved if depth information is also utilized while performing the fitting operation. Thus better datasets which have masked information as well will help in alleviating this problem. Also, further experimentation in the direction of complex shape parameters has the potential of yielding better results.

References

[1] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3d faces," 1999.

[2] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis, "Scape: shape completion and animation of people," in *ACM transactions on graphics (TOG)*, vol. 24, pp. 408–416, ACM, 2005.

[3] N. Hasler, C. Stoll, M. Sunkel, B. Rosenhahn, and H.-P. Seidel, "A statistical model of human pose and body shape," in *Computer graphics forum*, vol. 28, pp. 337–346, Wiley Online Library, 2009.

[4] L. Ballan, A. Taneja, J. Gall, L. Van Gool, and M. Pollefeys, "Motion capture of hands in action using discriminative salient points," in *European Conference on Computer Vision*, pp. 640–653, Springer, 2012.

[5] O. Freifeld and M. J. Black, "Lie bodies: A manifold representation of 3d human shape.," 2012.

[6] J. Taylor, R. Stebbing, V. Ramakrishna, C. Keskin, J. Shotton, S. Izadi, A. Hertzmann, and A. Fitzgibbon, "User-specific hand modeling from monocular depth sequences," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 644–651, 2014.

[7] J. S. Supancic, G. Rogez, Y. Yang, J. Shotton, and D. Ramanan, "Depth-based hand pose estimation: data, methods, and challenges," in *Proceedings of the IEEE international conference on computer vision*, pp. 1868–1876, 2015.

[8] A. Tkach, A. Tagliasacchi, E. Remelli, M. Pauly, and A. Fitzgibbon, "Online generative model personalization for hand tracking," *ACM Trans. Graph.*, vol. 36, pp. 243:1–243:11, Nov. 2017.

[9] J. Taylor, L. Bordeaux, T. Cashman, B. Corish, C. Keskin, T. Sharp, E. Soto, D. Sweeney, J. Valentin, B. Luff, A. Topalian, E. Wood, S. Khamis, P. Kohli, S. Izadi, R. Banks, A. Fitzgibbon, and J. Shotton, "Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences," *ACM Trans. Graph.*, vol. 35, pp. 143:1–143:12, July 2016.

[10] J. P. C. Valentin, A. Dai, M. Nießner, P. Kohli, P. H. S. Torr, S. Izadi, and C. Keskin, "Learning to navigate the energy landscape," *CoRR*, vol. abs/1603.05772, 2016.

[11] A. Tkach, M. Pauly, and A. Tagliasacchi, "Sphere-meshes for real-time hand modeling and tracking," *ACM Trans. Graph.*, vol. 35, pp. 222:1–222:11, Nov. 2016.

[12] R. Poppe, "A survey on vision-based human action recognition," *Image and vision computing*, vol. 28, no. 6, pp. 976–990, 2010.

[13] F. Bogo, A. Kanazawa, C. Lassner, P. V. Gehler, J. Romero, and M. J. Black, "Keep it smpl: Automatic estimation of 3d human pose and shape from a single image," in *ECCV*, 2016.

[14] M. Oberweger, P. Wohlhart, and V. Lepetit, "Hands deep in deep learning for hand pose estimation," *CoRR*, vol. abs/1502.06807, 2015.

[15] S. Melax, L. Keselman, and S. Orsten, "Dynamics based 3d skeletal hand tracking," in *Proceedings of Graphics Interface 2013*, GI '13, (Toronto, Ont., Canada, Canada), pp. 63–70, Canadian Information Processing Society, 2013.

[16] M. Schröder, J. Maycock, H. Ritter, and M. Botsch, "Real-time hand tracking using synergistic inverse kinematics," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 5447–5454, IEEE, 2014.

[17] A. Tagliasacchi, M. Schroeder, A. Tkach, S. Bouaziz, M. Botsch, and M. Pauly, "Robust articulated-icp for real-time hand tracking," *Computer Graphics Forum (Symposium on Geometry Processing)*, vol. 34, no. 5, 2015.

[18] T. Sharp, C. Keskin, D. Robertson, J. Taylor, J. Shotton, D. Kim, C. Rhemann, I. Leichter, A. Vinnikov, Y. Wei, *et al.*, "Accurate, robust, and flexible real-time hand tracking," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pp. 3633–3642, ACM, 2015.

[19] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, "Efficient model-based 3d tracking of hand articulations using kinect.,"

[20] D. Tomè, C. Russell, and L. Agapito, "Lifting from the deep: Convolutional 3d pose estimation from a single image," *CoRR*, vol. abs/1701.00295, 2017.

[21] J. Tompson, M. Stein, Y. Lecun, and K. Perlin, "Real-time continuous pose recovery of human hands using convolutional networks," *ACM Transactions on Graphics (ToG)*, vol. 33, no. 5, p. 169, 2014.

[22] C. Qian, X. Sun, Y. Wei, X. Tang, and J. Sun, "Realtime and robust hand tracking from depth," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1106–1113, June 2014.

[23] J. Taylor, L. Bordeaux, T. Cashman, B. Corish, C. Keskin, T. Sharp, E. Soto, D. Sweeney, J. Valentin, B. Luff, *et al.*, "Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, p. 143, 2016.

[24] X. Zhou, Q. Wan, W. Zhang, X. Xue, and Y. Wei, "Model-based deep hand pose estimation," *CoRR*, vol. abs/1606.06854, 2016.

[25] E. Dibra, T. Wolf, A. C. Öztireli, and M. H. Gross, "How to refine 3d hand pose estimation from unlabelled depth data ?," in *Fifth International Conference on 3D Vision (3DV), Qingdao, China, October 10-12, 2017*, 2017.

[26] C. Keskin, F. Kıraç, Y. E. Kara, and L. Akarun, "Hand pose estimation and hand shape classification using multi-layered randomized decision forests," in *European Conference on Computer Vision*, pp. 852–863, Springer, 2012.

[27] F. Mueller, F. Bernard, O. Sotnychenko, D. Mehta, S. Sridhar, D. Casas, and C. Theobalt, "Ganerated hands for real-time 3d hand tracking from monocular RGB," *CoRR*, vol. abs/1712.01057, 2017.

[28] Q. Ye, S. Yuan, and T.-K. Kim, "Spatial attention deep net with partial pso for hierarchical hybrid hand pose estimation," in *European conference on computer vision*, pp. 346–361, Springer, 2016.

[29] A. Sinha, C. Choi, and K. Ramani, "Deephand: Robust hand pose estimation by completing a matrix imputed with deep features," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4150–4158, 2016.

[30] M. de La Gorce, D. J. Fleet, and N. Paragios, "Model-based 3d hand pose estimation from monocular video," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 9, pp. 1793–1805, 2011.

[31] D. Tang, T.-H. Yu, and T.-K. Kim, "Real-time articulated hand pose estimation using semi-supervised transductive regression forests," in *Proceedings of the IEEE international conference on computer vision*, pp. 3224–3231, 2013.

[32] J. Zhang, J. Jiao, M. Chen, L. Qu, X. Xu, and Q. Yang, "3d hand pose tracking and estimation using stereo matching," *arXiv preprint arXiv:1610.07214*, 2016.

[33] T. Simon, H. Joo, I. A. Matthews, and Y. Sheikh, "Hand keypoint detection in single images using multiview bootstrapping.,"

[34] S. Sridhar, H. Rhodin, H.-P. Seidel, A. Oulasvirta, and C. Theobalt, "Real-time hand tracking using a sum of anisotropic gaussians model," in *3D Vision (3DV), 2014 2nd International Conference on*, vol. 1, pp. 319–326, IEEE, 2014.

[35] S. Sridhar, A. Oulasvirta, and C. Theobalt, "Interactive markerless articulated hand motion tracking using rgb and depth data," in *Proceedings of the IEEE international conference on computer vision*, pp. 2456–2463, 2013.

[36] V. Athitsos and S. Sclaroff, "Estimating 3d hand pose from a cluttered image," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 2, pp. II–432, IEEE, 2003.

[37] C. Zimmermann and T. Brox, "Learning to estimate 3d hand pose from single rgb images,"

[38] A. Spurr, J. Song, S. Park, and O. Hilliges, "Cross-modal deep variational hand pose estimation," *CoRR*, vol. abs/1803.11404, 2018.

[39] B. Allen, B. Curless, and Z. Popovi, "The space of human body shapes: reconstruction and parameterization from range scans," *ACM transactions on graphics (TOG)*, 2003.

[40] S. Khamis, J. Taylor, J. Shotton, C. Keskin, S. Izadi, and A. Fitzgibbon, "Learning an efficient model of hand shape variation from depth images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2540–2548, 2015.

[41] S. Yuan, Q. Ye, B. Stenger, S. Jain, and T.-K. Kim, "Bighand2. 2m benchmark: Hand pose dataset and state of the art analysis," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pp. 2605–2613, IEEE, 2017.

[42] C. Qian, X. Sun, Y. Wei, X. Tang, and J. Sun, "Realtime and robust hand tracking from depth," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1106–1113, 2014.

[43] Wikipedia contributors, "Hill climbing — Wikipedia, the free encyclopedia," 2019. [Online; accessed 4-May-2019].

[44] M. Rad, M. Oberweger, and V. Lepetit, "Feature mapping for learning fast and accurate 3d pose inference from synthetic images," *CoRR*, vol. abs/1712.03904, 2017.

Chapter 6

Appendix

This Chapter contains the portion of the work on Joints Net which is done by Pratik but is used in Shape estimation task to either improve the performance of ShapeNet or for jointly training the two network. Similarly this chapter also discusses the portions where ShapeNet is used to improve the performance of JointsNet. These two networks together are used to solve the problem of hand pose estimation. (See figure 6.1) The chapter ends with explaining the two different kinematics module - Forward Kinematics and Inverse Kinematics which are initially written by me in python and further implemented by Pratik in Tensorflow.

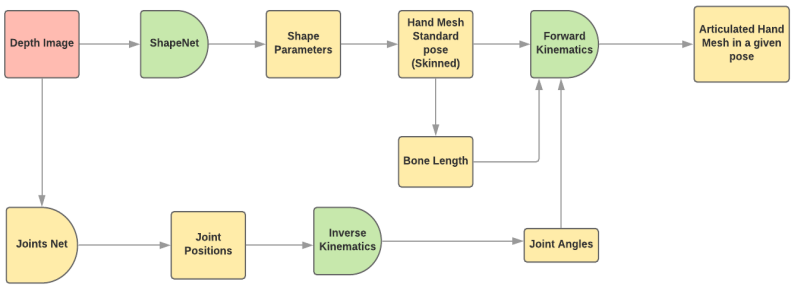


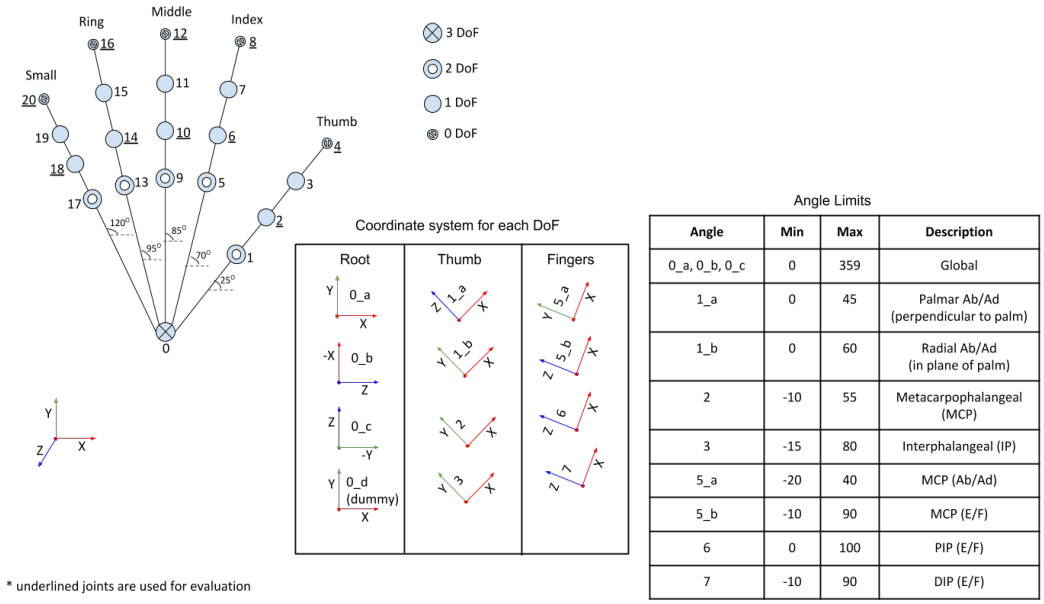
Figure 6.1: **Overall pipeline for Hand Pose estimation.**

6.1 Joint Position Estimation

Joint Position estimation refers to the task of estimating the joint positions of the hand in world coordinate system given the depth image of the hand. Similar to the Shape Estimation we tackle this problem using CNN based structure which takes depth images as input and predicts the joints position. The hand model has 21 different joints which we are predicting. One root joint at palm and 4 joints on every finger that we are predicting. See Figure 6.2 for further details related to the positioning of the joints.

6.1.1 JointsNet

The architecture of JointsNet is similar to the architecture of ShapeNet except that the last estimation layer has 21 X 3 = 63 as output instead of 4. Rest of the network structure is entirely the same. JointsNet also uses the same training procedure which is used by



Angle Limits

Angle	Min	Max	Description
0_a, 0_b, 0_c	0	359	Global
1_a	0	45	Palmar Ab/Ad (perpendicular to palm)
1_b	0	60	Radial Ab/Ad (in plane of palm)
2	-10	55	Metacarpophalangeal (MCP)
3	-15	80	Interphalangeal (IP)
5_a	-20	40	MCP (Ab/Ad)
5_b	-10	90	MCP (E/F)
6	0	100	PIP (E/F)
7	-10	90	DIP (E/F)

Figure 6.2: **Hand Model**

ShapeNet and is trained on BigHand dataset. The loss function is also the same, with the only difference being that instead of using MSE between shape parameters it computes MSE between the ground truth joint position and predicted joint position. Once the training is completed, JointsNet can then predict joint positions of the hand given the depth image. The joint positions are then passed to the inverse kinematics layer that compute the joint angles. These joint angles constitutes the pose of the hand that is currently in and is used by Forward Kinematics (refer Section 6.2) to transform the mesh from standard pose to the articulated pose.

6.1.2 Combined Training

Once both the networks ShapeNet and JointsNet are trained individually, we then train both of them together to improve their performance. The loss used for this purpose constitutes of two part. The first part is the MSE loss between the predicted joints position from JointsNet and ground truth joints from dataset. The second loss function acts as a kinematics based regularizer. We pass the predicted joint positions to the progressive IK function (refer Section 6.3) which computes the joint angles for each of the joints. These joints angles along with the bone length predicted by ShapeNet are then passed to the Forward Kinematics function which computes the joint positions. Thus, we use compute the MSE loss between these kinematics based joint positions and the ground truth joint positions to represent the second component of our combined loss.

$$L_{combined} = L_{jointsNet} + L_{Kinematics}$$

$$L_{jointsNet} = \sum_{i \in dataset} |GJP(i) - PJP(i)|^2$$

$$L_{combined} = \sum_{i \in dataset} |GJP(i) - FK(IK(PJP(i)), BL)|^2$$

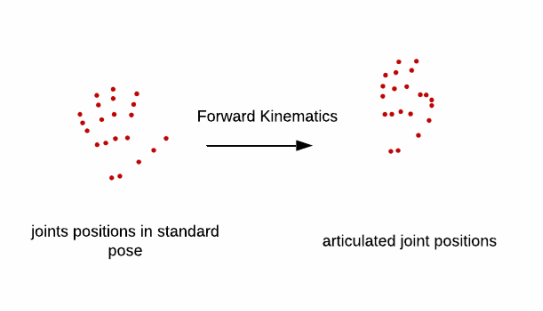


Figure 6.3: **FK on joint positions**

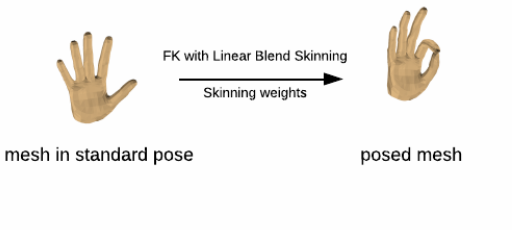


Figure 6.4: **FK using LBS**

where i is the index of element in the dataset, GJP is the ground truth joint position, PJP is the predicted joint position , BL is the bone lengths predicted by the ShapeNet , FK is the Forward Kinematics function and IK is the inverse kinematics function.

6.2 Forward Kinematics

Forward kinematics function refers to transforming the joint positions or mesh which are present in standard pose to the articulated pose. For transforming the joint positions we use the angles to compute the matrices for every joint in the hand. These matrices are computed by considering every finger as an independent kinematic chain starting from the root position in the palm and ending at the tip of the finger. (Refer figure 6.2). Thus by applying these transformation matrices to every joint position we thus obtain the posed joint positions. Similarly, for mesh in standard pose we first obtain the skinning matrix (refer section 3.6.1) and then use Linear Blend Skinning to transform every vertex using simple matrix multiplications. Figure 6.3 and 6.4 demsontrates the functionality of our FK function.

6.3 Inverse Kinematics

Inverse Kinematics refers to the operation of obtaining the angles from the vertex positions given the knowledge of the kinematic chain the vertices are part of. Since our hand model consists of 5 different kinematics chain represented by each of the fingers with starting position being the root of the palm and ending position being the tip of the finger. Since we know the hand model and the axes about which different joint moves (See figure 6.2) we thus obtain a close form solution of joints angles for every joint present in the hand.