Reconstructing NBA Players

1st Ajay Vasisht
Department of Computer Science: CS231A
Stanford University
avasisht@stanford.edu

***Abstract*—This report aims to recreate and implement an existing open-sourced methodology used for 3D reconstruction of NBA players, proposed in 2020 by researches at the University of Washington:** $https://arxiv.org/pdf/2007.13303.pdf$**. The problem is essentially single-view 3D reconstruction and pose estimation of NBA players from 2D images. While previous approaches struggle with modeling the challenging body poses, modeling clothing, and overcoming occlusion that exist in the domain of basketball activities, the 2020 paper introduces a pipeline approach that first learns pose and jump estimations from a 2D image, then calibrates for the NBA game-time camera, and the lastly outputs a high resolution mesh and 3D pose for a player. The secret sauce is use of NBA 2K video game models as training data to help learn the high resolution mesh. This report improves on some of the quantitative metrics using tuning hyper-parameters, but also confirms that the paper's code does give a way to model clothing shadow and molds in producing the final reconstruction.**

I. Introduction

Given a broadcast view of an NBA basketball, we'd like to complete a 3D reconstruction of the players, to be viewed from any camera view and angle. The reconstruction problem is difficult in basketball games because players shoot and run and jump in complex, nuanced poses and be occluded either by other players on the basketball court or by themselves.

The problem is addressed through a pipeline approach presented in the 2020 paper [1]. First, given a set of 2D $256x256$ images, we learn the 2D pose, 3D pose, jump classification and jump height. Next, the camera is calibrated to learn the camera parameters using correspondence points against a birds-eye view blueprint of a standard basketball court. Since the lines of the court are difficult to capture due to occlusion from players and referees, they are approximated with an optimization approach using those correspondence points. Then, to help with the 3D reconstruction, the authors of this paper procure a dataset of 3D models from the NBA 2K video game to capture diverse basketball-specific poses to train a mesh generation model on each body part of the NBA player, and then overlay the learned 3D pose onto the combined body parts to get the 3D reconstruction of the NBA player.

II. Background/Related Work

The problem of 3D reconstruction for sports has been explored by [2] and [3] both use multi-view stereo methods. Additionally, [4] has worked on reconstructing players from soccer games but they only predict depth maps and assume all players are on the ground, which is not a suitable assumption for the game of basketball. The 2020 paper [1] builds a reconstruction algorithm that accounts for complex airborne poses.

[4] also used FIFA soccer video game data to capture depth maps of soccer players, showing related work around using data from video game sot reduce manual labor and labeling bu capture ground-truth data while playing the game. The NBA 2K dataset used in this pipeline approach however decides to capture 3D triangle meshes, the 3D pose, and texture maps. More detail will be covered in the next section on how this dataset was procured.

There are many datasets available to have previously done to enable progress in single-view 3D human pose estimation from single images such as [5]. However, they do not assist to predicting global position and jump height, as is needed to fully reconstruct an NBA player in the basketball scene.

Most parametric models, including neural networks, also try to directly regress body shape parameters, and reconstruct undressed humans since clothing is not part of the parametric model, as seen in [6]. Some work has been in done in predefined garment models [7] to estimate a clothing mesh layer. However, [1] is able to incorporate the nuance of clothing with one neural net they trained specifically focusing on adding the clothes to each body part.

Non-parametric methods proposed voxel representations to model humans by training on synthetic data [8]. This project however focused on learning parameters for a new neural network they calling SkinningNet for the purpose of learning the 3D body with a clothing overlay.

Of course, this project aims to recreate [1], which focuses on single-view reconstruction of players to produce a 3D model of player pose and shape. The reconstruction needs to capture the complexity of basketball poses such as reconstructing jumps, dunks, dribbles, and shooting. As we'll see in the next section, the use of two different datasets - one of 2D images of basketball players for pose estimation and camera calibration, and one of 3D models in complex poses from the 2k basketball video game - allow for their paper to show great results.

Unlike prior methods of modeling undressed people in various poses of dressed people in a frontal pose, the 2020 paper [1] focuses on modeling clothed people under challenging basketball poses.

III. Datasets

The first dataset used is a standard set of 2D $256x256$ images of basketball scenes from a broadcast view. The

authors provide a dataset of broadcast view. These datasets will be pre-processed using OpenPose as an object and keypoint detection model to capture bounding boxes for each player, then from those bounding boxes they would crop transform and center and scale the relevant player to a $256x256$ image.

The second and most important dataset is the NBA 2K players samples from the NBA 2K video game from 2019, known as 2K19. These models are extremely detailed and realstic down to the wrinkles and ruffles in clothing. The 2020 paper [1] captures 27,144 basketball poses across 27 NBA players and contains body mesh and texture data collected by the researchers playing the NBA2K19 game and intercepting calls between the game engine and the graphics card.

The program captures all drawing events per frame, where the authors locate player rendering events by analyzing the hashing code of both vertex and pixel shaders. Next, triangle meshes and textures are extracted by reverse-engineering the compiled code of the vertex shader. The game engine renders players by body parts, so they perform a nearest neighbor clustering to decide which body part belongs to which player. Since the game engine optimizes the mesh for real-time rendering, the extracted meshes have different mesh topologies, making them harder to use in a learning framework. So, they register the meshes by resampling vertices in texture space based on a template mesh. They also extract the rest pose skeleton and per-bone transformation matrix, from which to compute forward kinematics to get full 3D pose.

It is important to note this dataset is used for the pose estimation for which they use a modified PoseNet. This training requires generalization on real images. Thus, they augment the data to 65,765 training examples, 37,966 validation examples, and 66,442 testing examples. Augmentation for synthetic data is done by rendering and blending meshes into various random basketball courts to produced player-centered $256x256$ 2D images in .png format. This means the dataset was trained with real images from NBA games but also synthetic data from the video game. Fortunately, all of this data was provided by the authors already pre-processed.

The authors point out that due to privacy and non-approval from the 2K corporation, they could not publish the 2K dataset they used for their paper. They instead provide a dataset from base, generic players with the same statistics as the NBA players in the game. This means the models I trained will be expected to have different results quantitatively as well as qualitatively. And in fact, metrics around pose estimation and the mesh reconstruction metrics did deviate by a small amount.

IV. Approach

We can break down the pipeline approach into 3 main steps: the pose and jump estimation, the camera calibration, and the mesh generation.

Above is a diagram illustrating each component.

A. Pose and Jump estimation

The dataset of 2D images do not contain skeletal information, so the authors propose a neural net they call PoseNet

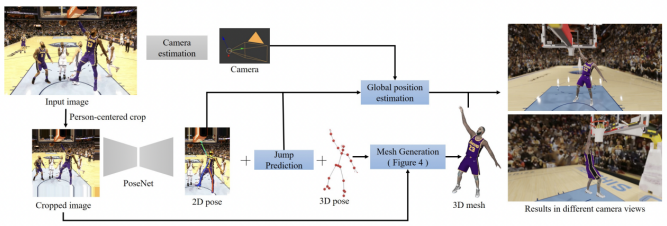

Fig. 1. Full Pipeline Diagram

to estimate the 3D pose, 2D pose, jump height, and jump classification of a player from a single image.

The architecture consists of a backbone of ResNet from which to extract features and then supply them to 4 separate branch heads. One branch predicts 2D pose and outputs 2D heatmaps for each keypoint to show where the keypoint is located. Another predicts 3D pose and outputs XYZ location maps for each keypoint. For these branches, it's a small network of 3 blocks of Deconvolution, Batch Normalization, and ReLU blocks. To obtain the final output, the location of the maximum value in every keypoint heatmap is used to get the 2D pose at 64 × 64 resolution and use it to sample the 3D pose from the XYZ location maps. Then , the 2D pose is transformed to original $256x256$ resolution.

The next two branches are a classification and regression branch for prediction of jump height. The authors use a fully connected layer then two linear residual blocks to get the output. The authors estimate both the jump class and the jump height because the jump class can serve as a threshold to reject the inaccurate jump height prediction in the global position estimation. The ground truth jump height is directly extracted from the game, and the jump class is set to 1 if the jump height is greater than 0.1m.

The loss is calculated as followed through forward propagation, and is optimized using Adam optimization for the backpropagation step to updated the weights.

$$L_{pose} = w_{2d}L_{2d}+w_{3d}L_{3d}+w_{height}L_{height}+w_{jumpcls}L_{jumpcls} \quad (1)$$

Breaking it down, we calculate the loss as a weighted sum of the loss from each branch. Each regression branch loss (2d pose, 3d pose, and jump height) are all L1 loss (mean absolute error), and the classification branch is Cross Entropy Loss.

Below are the default training parameters used for the model.

See figures 2 and 3 for an example of how this step looks.

B. Camera Calibration for Global Positioning

To estimate the global position of the player we need the camera parameters of the input image. Since NBA courts have known dimensions, calibration is done by taking a bird's eye blueprint image of an NBA court and generating a synthetic field aligned with input frame. Then, manually the user can input 4 correspondences between the input image and bird's

TABLE I
POSENET PARAMETERS

Variable	Value
Batch Size	16
Valid Batch Size	16
Number of Epochs	100
Decay Gamma	0.00005
Log Interval	20
Learning Rate	0.001
w_{2d}	10
w_{3d}	10
w_{height}	0.4
$w_{heightcls}$	0.2

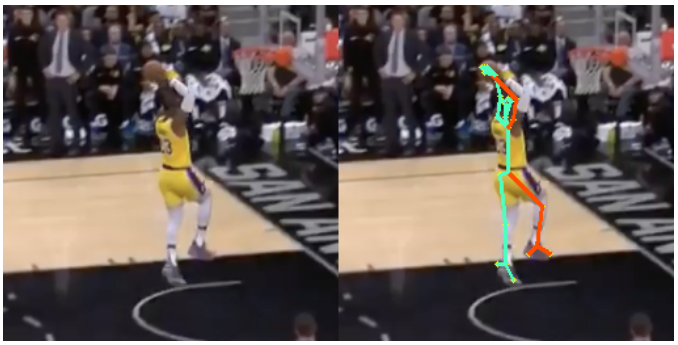

Fig. 2. Lebron: 2D pose estimation

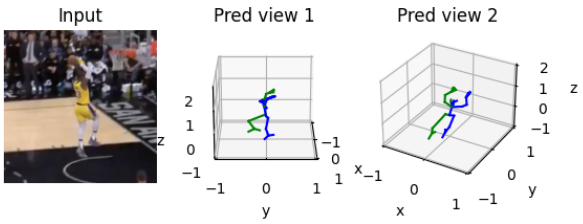

Fig. 3. Lebron: 3D pose estimation

eye blueprint image to initialize the camera parameters. The authors give a line-based optimization technique to further refine the calibration but I found it was not necessary for the tests I did.

To get robust line features, we train a pix2pix network to translate basketball images to court line masks. For the training data, we use synthetic data from NBA2K, where the predefined 3D court lines are projected to image space using the extracted camera parameters.

After estimating the camera parameters, we place the player mesh in 3D by considering its 2D pose in the image and the jumping height. We can use the following equations for casting a ray from the center of the 3D court through the lowest keypoint of the player:

$$V_c = \begin{bmatrix} (x_p - p_x) * z_c/f \\ (x_y - p_y) * z_c/f \\ z_c \end{bmatrix} \tag{2}$$

$$[y_w = R_2 * (v_c - T)] \tag{3}$$

R_2 is the second column of the extrinsic rotation matrix and T is the extrinsic translation, while f is the focal length. p_x and p_y are the principle point namely the center of the 3D court. y_w is the world coordinate y-component of the lowest joint which is the predicted jump height. V_c are the camera coordinates of the lowest joint namely the foot. x_p and y_p are the camera coordinates of the lowest joints. We can easily calculate z_c which is the camera coordinate z-component of the lowest joint to fully compute the global position of the player using the above system of equations.

In sum, this part uses correspondences between the broadcast view and the birds-eye view of a basketball court to calibrate the camera, and uses a neural network called pix2pix to translate basketball images to court line masks. Both of these along with the jump height from the previous section are used to calculate the global position of the player.

See figures 4, 5 and 6 for an example of how this step looks.

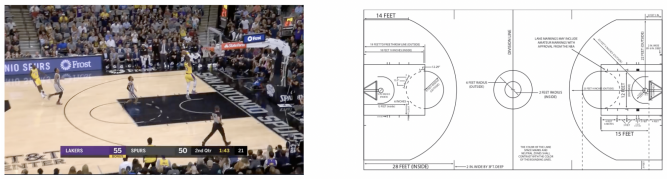

Fig. 4. Manually Generating Correspondences

C. Mesh Generation

Mesh Generation is done through two sub nets: Identity Net and Skinning Net (Diagram in Fig 7).

The Identity net deforms a rest posh template mesh (average pose over all players in the database) into a personalized rest pose mesh. The main benefit of this network is that it allows us

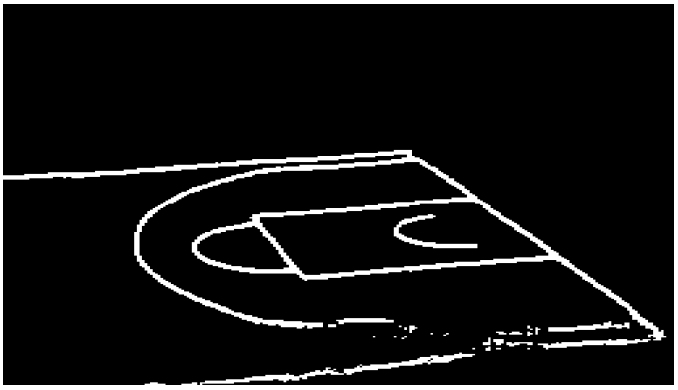

Fig. 5. Court Line Masks after pix2pix using Correspondences

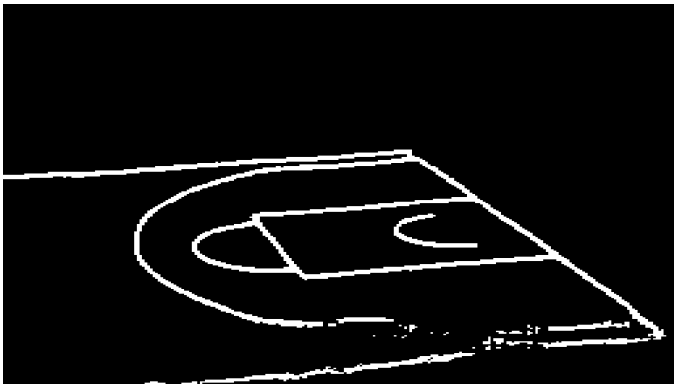

Fig. 6. Line Masks Overlayed onto original image

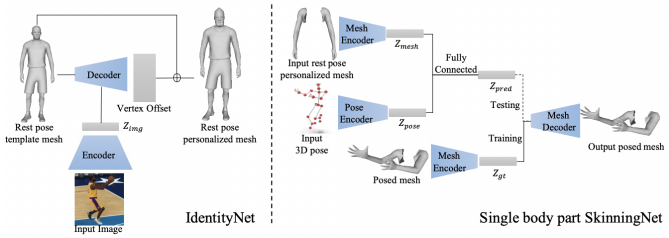

Fig. 7. Mesh Generation

to estimate the body size and arm span of the player according to the input image.

For Identity Net, ResNet is used as the backbone to extract features from input images. Then the template mesh vertices with image features are concatenated and sent to a decoder called AtlasNet to predict vertex offsets. The offset is then added to the template mesh to get the personalized mesh. L1 loss is used to train the Identity Net. In sum, this net learns the delta needed from template mesh to create the personalized mesh.

The Skinning Net takes this personalized mesh in rest pose and 3D pose outputted from the first step to output the posed mesh. To reduce the learning complexity, the meshes are pre-segmented into six parts: head, arms, shirt, pants, legs and shoes and train a Skinning Net on each part. Finally, the six reconstructed parts are combined into one, while removing interpenetration of garments with body parts. This segmentation is helpful also because the NBA 2K training is also given by body part.

The SkinningNet architecture is a TL-embedding network to learn latent spaces: Z_{pose} by encoding the 3D keypoints found from Pose Estimation and Z_{rest} by separately encoding the the personalized meshes from the Identity Network. Both latent vectors are concatenated and fed into a full connected layer to get Z_{pred}. Meanwhile the ground truth vertices are fed into a different mesh encoded to get Z_{gt} on which the loss is calculated against. And then, the Z_{pred} to fed to a decoded to get V_{pred} to compare against the original ground truth V_{posed}.

The Pose encoder is comprised of two linear residual blocks (fully connected, batch normaliation, ReLu, Dropout - with skip connection) followed by a fully connected layer. The mesh encoder consists of four SC-ELU blocks then a fully connected layer. The mesh (shared) decoder are built with spiral convolution blocks then upsampling and then a last spiral convolution layer.

The loss for Skinning Net is calculated as followed through forward propagation, and is optimized using Adam optimization for the back-propagation step to updated the weights.

$$L_{skin} = w_z L_z + w_{mesh} L_{mesh} \tag{4}$$

Breaking it down, we calculate the loss as a weighted sum of the loss from the latent space L1 Loss and the actual decoded meshes. The weights determine how much to attribute loss to the encoded latent space differences versus the actual decoded meshes against the actual meshes. Namely, $L_Z = ||Z_{pred} - Z_{gt}||$ and $L_{mesh} = ||V_{pred} - V_{posed}||$.

Below are the default training parameters used for the models. It's important to note that the authors report a learning rate of 0.0002 for the Identity Net whereas in their code and what I trained on, that default was 0.00003.

TABLE II
IDENTITY NET PARAMETERS

Batch Size	16
Valid Batch Size	16
Number of Epochs	200
Decay Rate	0.99
Decay Steps	1
Weight Decay	0.00005
Log Interval	20
Learning Rate Identity Net	0.0003

D. Combining Body Parts

Lastly, the body parts cannot just be concatenating as they will collide, do the meshes need to be deformed at the ends by moving collision vertices inside the clothes while preserving rigidity. This is an iterative process until no collisions or exceeding a threshold.

TABLE III
SKINNING NET PARAMETERS

Variable	Value
Batch Size	16
Valid Batch Size	16
Number of Epochs	200
Decay Rate	0.99
Decay Steps	1
Weight Decay	0.00005
Log Interval	20
Learning Rate Skinning Net	0.001
w_Z	5
w_{mesh}	50

V. EXPERIMENTATION: QUANTITATIVE

The Pose Estimation metric calculated on the test set provided by the authors is the mean per-joint position error. This is the average Euclidean distance between ground truth and prediction for a joint across all joints. The paper also reports the MPJPE after Procrustes Alignment, which is MPJPE calculated after the estimated 3D pose is aligned to the groundtruth by the Procrustes method, a similarity transformation.

TABLE IV
POSE ESTIMATION METRICS

Variable	Value
MPJPE	65.99111
MPJPE-PA	44.62177

This actually deviates quite a bit from the reported $MPJPE = 51.67$ and $MPJPE - PA = 40.91$, and we can attribute this from the fact that the provided dataset is different from the NBA 2K dataset because they were not able to release the original dataset of real NBA players, and had to duplicate players skillsets and abilities over to default characters to recreate the dataset I used. Naturally, this would result in differences quantitatively. Here, the paper's results are better than my experiments.

The Mesh Reconstruction metrics calculated on the test set provided by the authors are the Chamfer Distance scaled by 1000 as well as Earth-mover distance.

TABLE V
MESH RECONSTRUCTION METRICS

Variable	Value
CD	2.85369
EMD	0.07282

This actually deviates quite a bit from the reported $CD = 4.9834$ and $EMD = 0.087$, and we can attribute this from the fact that the provided dataset is different from the NBA 2K dataset. Surprisingly, these metrics are better than the reported ones, which is to say the mesh reconstruction trained on the default 2K datasets beats the mesh reconstruction trained on the 2K dataset with actual players. Here, the paper's results are worse than my experiments.

This made me consider improving the pose estimation by tuning hyperparameters, specially I was interested in re-weighting the loss function to incorporate more loss from the 3d pose loss, as this would be directly responsible in calculating $MPJPE$. I'm keeping the 2d pose weight the same.

TABLE VI
POSENET PARAMETERS

Variable	Value
Batch Size	16
Valid Batch Size	16
Number of Epochs	100
Decay Gamma	0.00005
Log Interval	20
Learning Rate	0.001
w_{2d}	10
w_{3d}	20
w_{height}	0.4
$w_{heightcls}$	0.2

I end up getting the following improvements in Pose Estimation Metrics. These metrics are more inline with the paper's results and beat the default metrics.

TABLE VII
METRICS IN 2ND EXPERIMENT

Variable	Value
MPJPE	53.15732
MPJPE-PA	41.9822

Seeing how these new 3d pose predictions impact the mesh generation, below are the metrics for that:

TABLE VIII
METRICS IN 2ND EXPERIMENT

Variable	Value
CD	2.83129
EMD	0.07396

Surprisingly, these metrics stay around the same even with seemingly better 3d poses. My assumption is that in the encoder places more of an emphasis on Z_{mesh} instead of Z_{pose} when both are concatenated, so improving 3d poses marginally does not making a big different later in the pipeline.

VI. EXPERIMENTATION: QUALITATIVE

Below are some samples of reconstructions compared to the original images using the default settings in the paper (before my 2nd experiment). I used Meshlab to render the 3D images from .obj files.

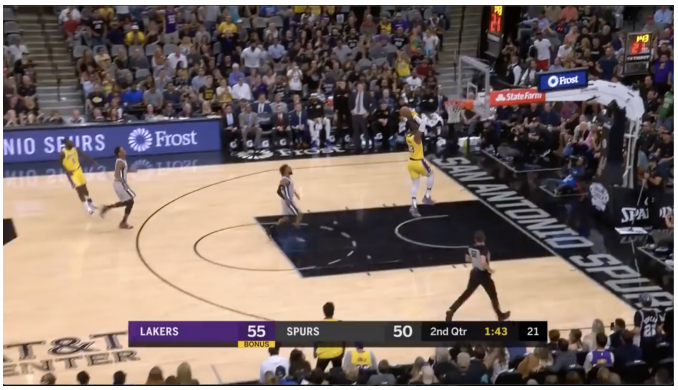

Fig. 8. Lebron: Original Image

Fig. 9. Lebron: 3D mesh construction angle 1

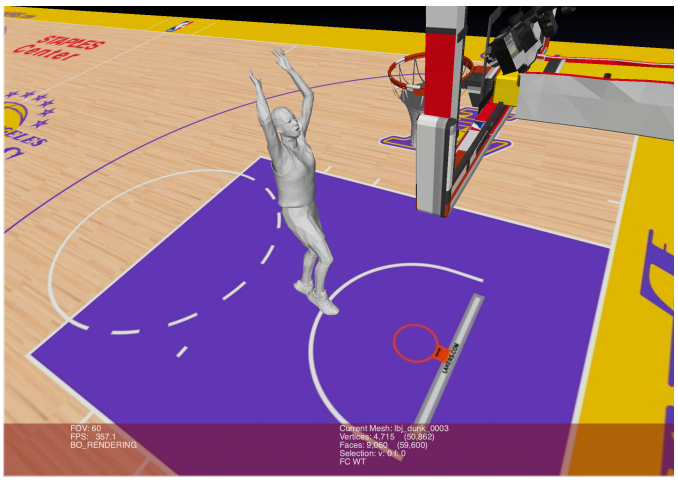

Fig. 10. Lebron: 3D mesh construction angle 2

Fig. 11. Lebron: 3D mesh construction angle 3

Fig. 12. Lebron: 3D mesh construction angle 4

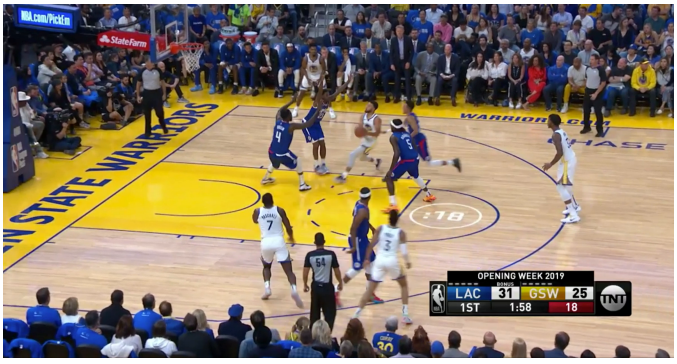

Fig. 13. Curry: Original Image

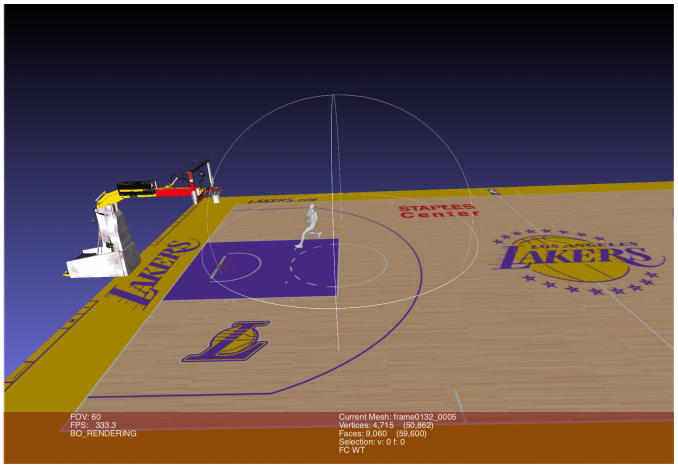

Fig. 14. Curry: 3D mesh construction angle 1

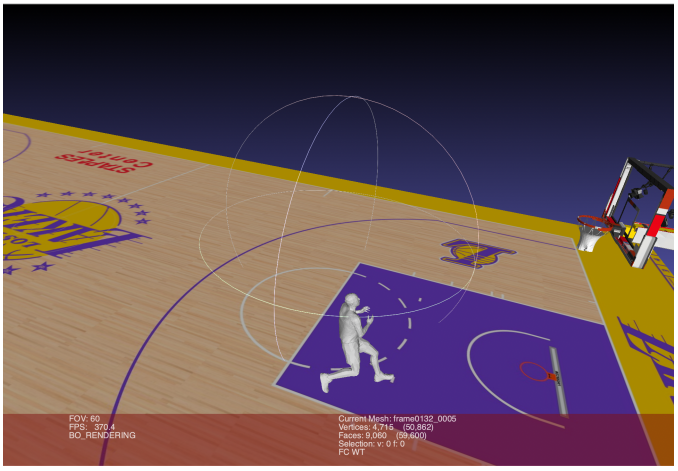

Fig. 15. Curry: 3D mesh construction angle 2

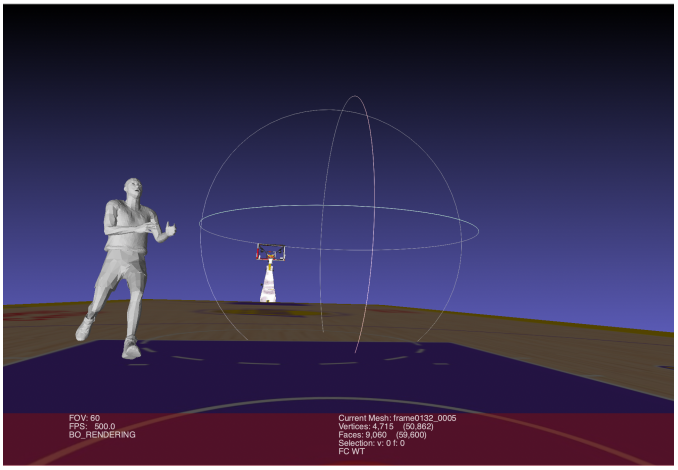

Fig. 16. Curry: 3D mesh construction angle 3

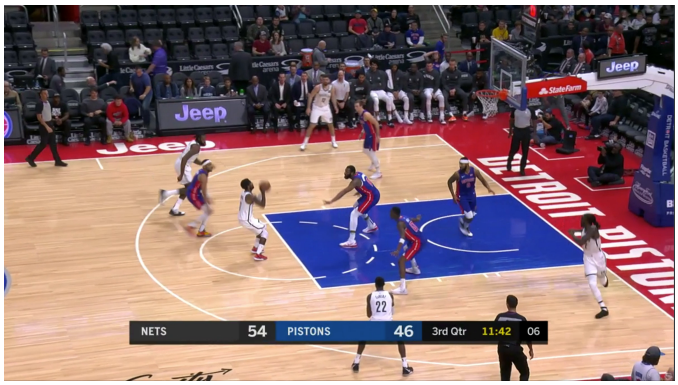

Fig. 17. Kyrie: Original Image

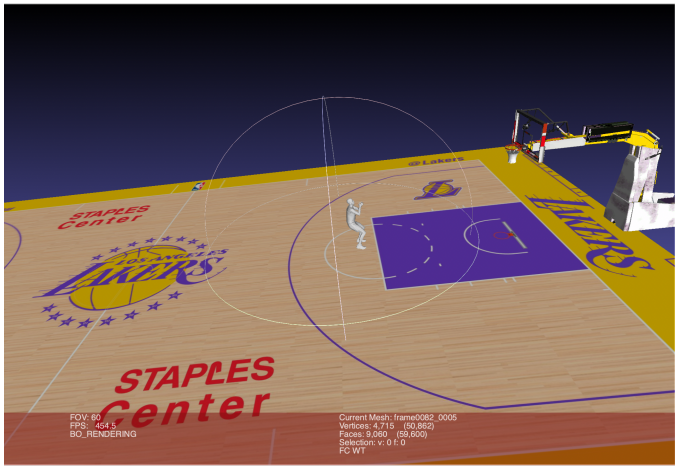

Fig. 18. Kyrie: 3D mesh construction angle 1

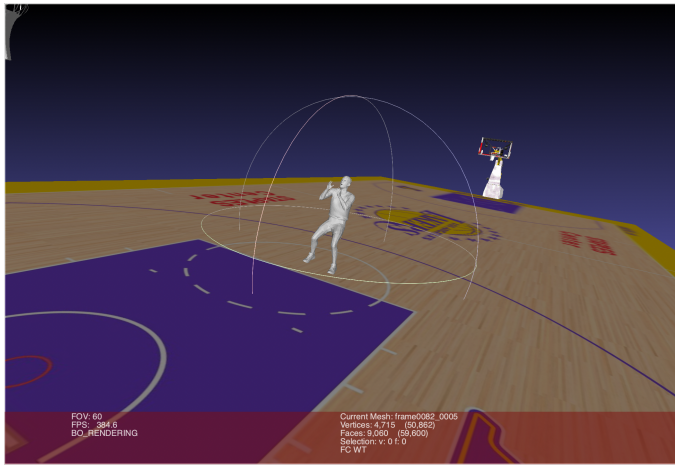

Fig. 19. Kyrie: 3D mesh construction angle 2

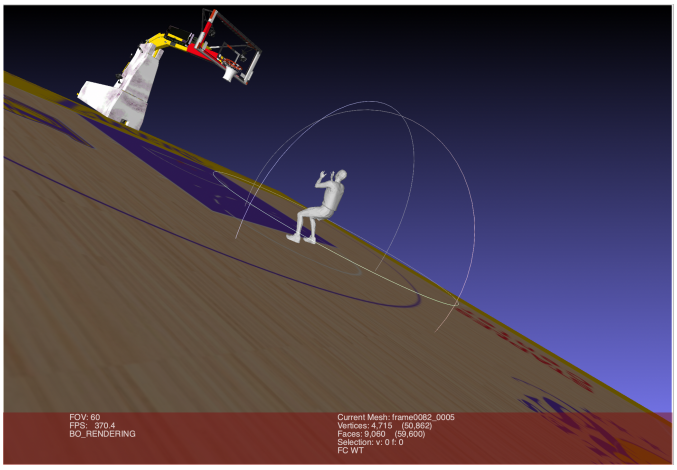

Fig. 20. Kyrie: 3D mesh construction angle 3

VII. Conclusions and Future Consideration

I found this pipeline approach to be a great connection between geometric and learning based approaches to computer vision. The results still struggle with occlusion, as well as modeling hair and facial identity due to lack of details in the low resolution input images. An area to improve would be to incorporate temporal dynamics on video instead of an image.

A-NERF: $https://arxiv.org/pdf/2102.06199.pdf$ is the most obvious next step for state-of-the-art 3D reconstruction. This particular version of NERF is to tuned to learn human poses and shape, and the examples looks like this model can capture the nuance of basketball poses. Had I had more time, I would tried to compare this paper's approach to what I implemented above.

VIII. Bibliography

References

[1] Zhu, Luyang and Rematas, Konstantinos and Curless, Brian and Seitz, Steve and Kemelmacher-Shlizerman, Ira, "Reconstructing NBA players," Proceedings of the European Conference on Computer Vision (ECCV) (2020)

[2] Grau, O., Thomas, G.A., Hilton, A., Kilner, J., Starck, J.: A robust free-viewpoint video system for sport scenes. In: 2007 3DTV conference. pp. 1–4. IEEE (2007)

[3] Guler, R.A., Kokkinos, I.: Holopose: Holistic 3d human reconstruction in-the-wild. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)

[4] Rematas, K., Kemelmacher-Shlizerman, I., Curless, B., Seitz, S.: Soccer on your tabletop. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4738–4747 (2018)

[5] Ionescu, C., Papava, D., Olaru, V., Sminchisescu, C.: Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. IEEE transactions on pattern analysis and machine intelligence 36(7), 1325–1339 (2013)

[6] Kolotouros, N., Pavlakos, G., Black, M.J., Daniilidis, K.: Learning to reconstruct 3d human pose and shape via model-fitting in the loop. In: Proceedings of the IEEE International Conference on Computer Vision (2019)

[7] Bhatnagar, B.L., Tiwari, G., Theobalt, C., Pons-Moll, G.: Multi-garment net: Learning to dress 3d people from images. In: IEEE International Conference on Computer Vision (ICCV). IEEE (oct 2019)

[8] Xiao, B., Wu, H., Wei, Y.: Simple baselines for human pose estimation and tracking. In: European Conference on Computer Vision (ECCV) (2018)