NBA 3D SCENE RECONSTRUCTION FROM MULTIPLE 2D RGB VIEWS

Akarsh Kumar, Kush Desai

University of Texas at Austin

ABSTRACT

The goal of our project was to create a complete 3-dimensional (3D) simulated view of a landscape from multiple 2-dimensional (2D) perspectives. Our project was specifically focused on reconstructing an NBA scene in 3D with the court, players, ball and cameras. In this project, we used raw video data from a highlight play in a 2020 NBA game and were able to successfully create a 3D depiction of the game. This project has applications in 3D play analysis, creating 3D datasets for basketball games, and eventually replacing referees with models trained to detect foul calls. Our main contribution for this field was the energy minimization technique that found the camera pose throughout the video for each view.

Index Terms— 3D scene reconstruction, 3D human pose, camera pose estimation

1. INTRODUCTION

In the second lecture of the semester, we learned that changing dimensions inherently loses valuable information. For example, taking a 2-dimensional (2D) picture of the 3-dimensional (3D) real-world throws away depth information. As such, pictorial and video representations from a single perspective cannot be considered accurate records of the scene. Therefore, to allow for a more complete reconstructions of a scene, humans tend to take pictures or record video simultaneously from multiple different perspectives. This multi-modal method of recording and presenting data allows a human viewer to mentally reconstruct the scene.

2. PROBLEM STATEMENT

Recently, 3D cameras and 3D technology has been popularized and commercialized. Many off-the-shelf solutions are available for mobile devices to record 360 degree or 3D pictures and video. Additionally, many companies sell 3D camera to record high quality 3D images and video. However, this technology cannot be used retroactively on old recordings or for high speed events.

In this project, we use video TV broadcasts from the 2020 NBA season (downloaded from YouTube). We specifically chose one game from the 2020 season since the games were played in an isolated "bubble" without fans. This minimized the noise in the background and allowed for simpler preprocessing. We identified three distinct viewpoints per game (broadcast, half-court and rim) and manually segmented and aligned the video.

Using the open-source software ffmpeg, we sampled each video from each frame at 15 frames per second (FPS). Thus, our dataset consisted of a sequence of frame images from each viewpoint, for each NBA game we had selected.

3. PRIOR WORK

Before getting into our specific methodology, we thought it was important to acknowledge the prior work done in this field and explain how this both differs from our work, yet impacted our mode of thinking.

We relied heavily on prior work by Dong et. al. regarding multi-person 3D Pose Estimation from Multiple Views. They focused on robustly recovering 3D poses for multiple people in a crowded region using multiple camera views [1]. Additionally, 3D geometry has also been exploited to combine input images into pose representations using multiple angles [2]. Finally, Burenius et. al. worked on extending a 2D pose estimation framework known as pictorial structures to a 3D landscape [3]. However, all these approaches rely on a priori knowledge or assumptions of the camera location and internals for each viewpoint. We attempted to experimentally determine the camera intrinsic and extrinsic data.

Work has been done on tracking players specifically in the basketball domain [4]. However, this focuses on single camera tracking with no mention of 3D reconstruction.

Additionally, Haenselmann, Busse and Kopf propose an algorithm which can stitch together views with different centers of projection into a single panoramic image [5]. This works primarily for images, however, and generates a panoramic representation of the work. Our work primarily looks at different viewpoints and takes into account relevant pose keypoints to generate a 3D viewpoint, not just a stitched panoramic image.

Thanks to Dr. Bovik and the TA team.

Fig. 1. AlphaPose output for a frame

Fig. 2. Different occurrences of basketballs found in our dataset

4. PLAYER POSE ESTIMATION IN 2D

Work on estimating human pose during motion in 3D in the wild has been done [6]. However, the problem of representing pose data for partially occluded images with motion noise remains unsolved.

We used AlphaPose to accurately generate pose information for the NBA domain [7] [8] [9]. This is a highly accurate (75 mAP) on COCO, open-source multi-person pose estimator released by the Machine Vision and Intelligence Group at the Shanghai Jiao Tong University. We opted to use AlphaPose over other open-source pose estimators like OpenPose, DeepCut and NRNet for multiple reasons. Firstly, AlphaPose comes with an online pose tracker called Pose Flow, which helps us track players across the court between frames. More importantly, AlphaPose works robustly with many occlusion cases, which is imperative in a closed, high-energy and high-motion environment like a NBA court. A sample output for a frame can be seen in Fig 1.

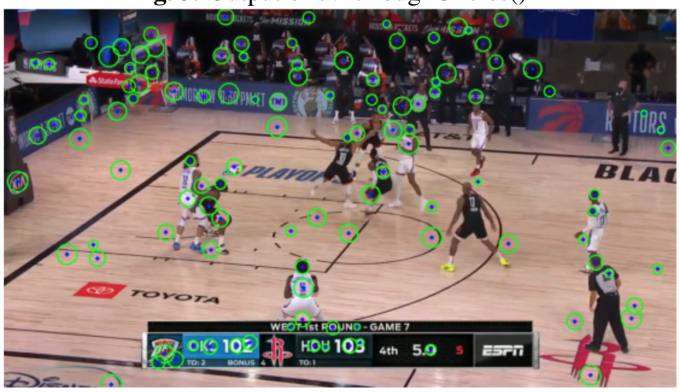
We ran a Fast Pose DUC model with a ResNet152 backbone and a YOLOv3 detector (from the Model Zoo). The actual pose detection was run on the NBA image frame dataset on an NVIDIA GeForce GTX 1070 Mobile GPU. This procedure converted the dataset by overlaying wireframe pose representations on every person in each frame.

5. BALL POSE ESTIMATION IN 2D

Ball tracking is a tricky problem, especially in this context. Most algorithms use a combination of geometric, visual color-based and deep learning features [4]. However, the basketballs in our dataset varied greatly in color and shape, making an objective ground truth "basketball" impossible to identify (see Fig. 2).

To start off, we tried a naive template matching algorithm. This consisted of using random selections of basketball images from the frames and running the cv2.matchTemplate() function on every subsequent frame. However, this method was abysmal at detecting the ball under different lighting conditions and different angles.

Fig. 3. Output of cv2.HoughCircles()



We then switched over to using the cv2.HoughCircles() method. This returned a sequence of various circles throughout the frame image, with many false positives (see Fig. 3).

To find the basketball in the frame, we first manually collected a dataset of 18 ground-truth basketballs from different views (Fig. 2). This served as the color dataset for finding the basketball.

To filter out false positives, we tried 3 different color correspondence metrics. First, we isolated the colors of the detected circle and found the L2 distance between these colors and the closest colors in the dataset.

To account for lighting issues, we also converted the colors to HSV and ignored the value component when performing L2 distance.

Another approach we did was to fit a Gaussian Mixture Model (GMM) to the color dataset and get a probability distribution over the color space. Then the detection was scored by performing a log likelihood of the colors being generated by the GMM probability distribution.

None of these techniques were perfect for detecting the ball, but we ended up using the GMM as it could generalize better to the colors of the ball.

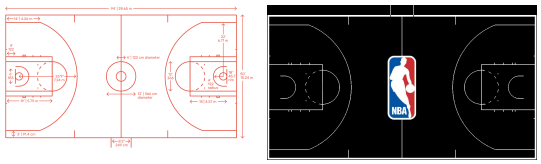
6. CAMERA POSE ESTIMATION

Finding the camera pose for each frame of each view is an essential step in reconstructing the scene. In order to accurately triangulate the 3d key points of the players and the ball, an accurate measurement of the camera focal length (camera intrinsics) and camera position and orientation (camera extrinsics) are essential. Three numbers are needed to store the camera position, and three are needed to store the camera orientation in an axis-angle representation. One number is needed to store the focal length. We refer to these seven numbers as the camera's degrees of freedom (DoFs). The goal of this section is to find these camera DoFs for each frame of each view.

6.1. Court model

A NBA course schematic was found online and heavily processed in GNU Image Manipulation Program (GIMP) to construct a model for the real NBA court seen in the data.

Fig. 4. Pre and Post Processed court schematic

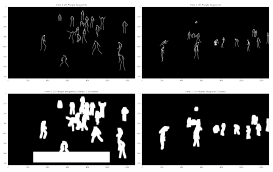


6.2. Image pre-processing

In order to use an image to find the camera pose, the data must be pre-processed in order to make the image look somewhat like the model and create a comparison metric. The goal of the pre-processing was to isolate only the court marking lines, as those are the prominent features in the model image.

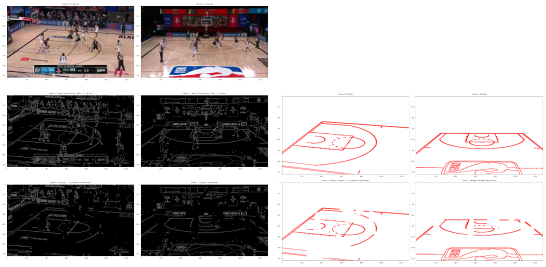
The first step was the find which parts of the court in the image was being occluded and could not be used for seeing court features. An occlusion mask was found by taking the player 2D models and drawing them onto a black image, then morphological dilating the image. The score-box was also manually added onto the occlusion mask.

Fig. 5. Court Occlusion Mask



Afterwards, multiple techniques were experimented with like edge detection (Sobel, Scharr, Laplacian), smoothing (gaussian, median, bilateral), adaptive and non-adaptive color thresh-holding, and Hough transform (find lines in the image). The technique we found to be most effective was to use an edge-preserving filter, conversion to gray-scale, followed by Canny edge detection. We also apply the same procedure to our model image before projecting it onto a camera plane.

Fig. 6. Data and model pre-processing stages. The data (left) goes from original, edge preserving filter+canny, occluded mask. The model (right) goes from original projection to occluded mask.



In order to get the camera pose for all the frames, the first frame must be computed, which is the initialization. Afterwards, the next frame could be calculated with an energy minimization algorithm initialized with the current frame. This could create a chain of optimizations each initializing the next one.

6.3. Camera Pose Initialization

Many methods for camera pose initialization were attempted. The simplest was to run the SIFT algorithm to calculate key points between the model image and the data image (preprocessed). This however did not work because there was too much noise in the data from background points and SIFT is not affine invariant (changes in perspective are too big). The ASIFT algorithm was also tested but did not prove to be useful.

One of our biggest ambitions was to use population based search over the camera pose space in order to find camera poses that accurately model the data seen from the first frame. In order to evaluate how good a solution was, a fitness/score was assigned to the solution based a pt2pt metric defined in the next section. Unfortunately, this population based search did not work due to the local minima of the loss landscape. Instead, we opted to hand initialize the first frame, and left the population based search tuning and loss function for future work. After manual labeling, we found a homography matrix to go from data image to 3D model court. From these correspondences it is also possible to find the camera rotation and translation using OpenCV's solution to the perspective N point problem. This however requires a knowledge of the intrinsics of the camera (focal length). A log space grid search was performed over the focal length to find the ideal match.

Fig. 7. Population based search over camera pose space. The bottom image are 16 views of the re-projected model court generated from sampling the distribution with the focal lengths written above.

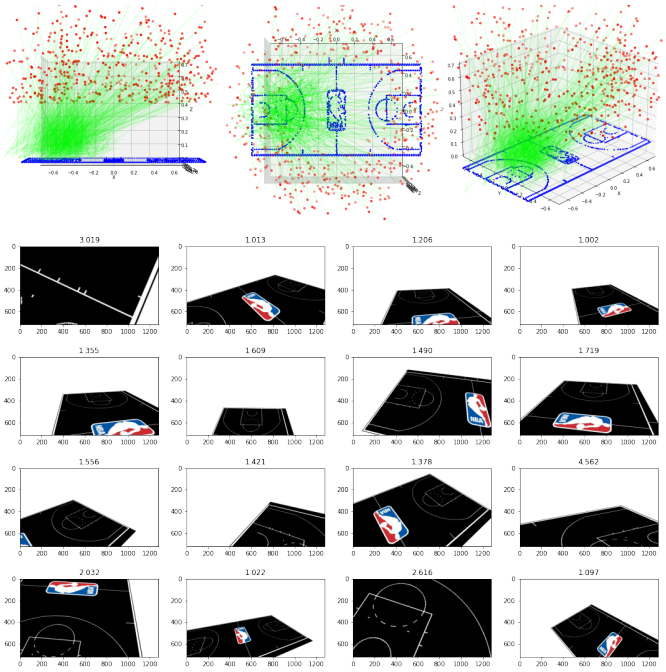
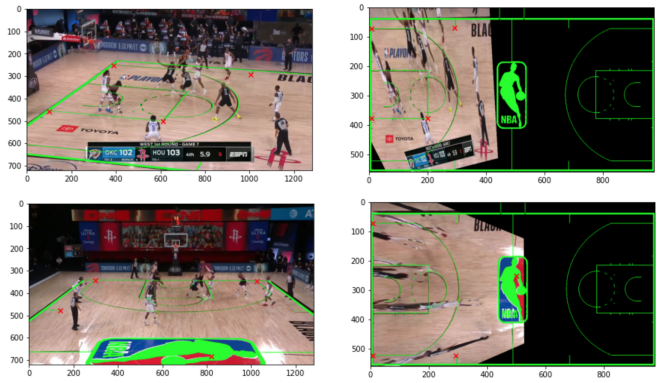


Fig. 8. Manually labeled four 2D key points in the image and four 3D key points in the model.

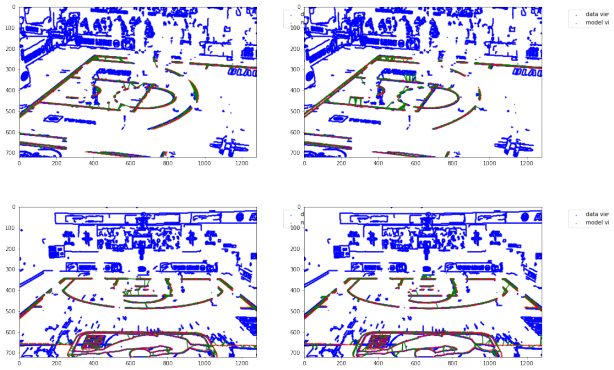


6.4. Camera pose optimization

This section covers the use of energy minimization to find the camera pose for the next frame given the previous frame.

We turned the model into a 3D point cloud which was projected onto the image plane. Similarly, the data image, after pre-processing, was turned into a 2D point cloud in the image plane. This entire projection pipeline was made end to end differentialable with PyTorch. A loss function called pt2pt was defined as the average distance of the point cloud of the model to the closest point in the point cloud of the data. A lower loss would mean the model more accurately models the data. After back-propagation through the pipeline all the way to the camera DoFs, the Adam optimizer was used to step the camera DoFs as to minimize the loss function. This procedure took in the previous frame's camera DoFs and estimated the current frame's camera DoFs for each frame for each view. This worked out very effectively.

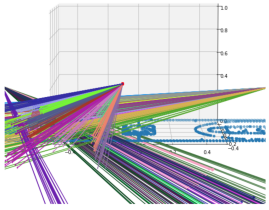
Fig. 9. Two frames visualization energy minimization. Blue points are the data point cloud. Red points are the model point cloud. Green vectors show pt2pt distance between randomly sampled points in the model and data.



7. 3D TRIANGULATION

Essentially, 3D triangulation focused on finding the 3D locations of key-points given the 2D key-point locations in the image plane of multiple views. The easiest way to understand this is visualizing a series of rays shooting out of each camera centers and through the pixel of the 2D key-point (Fig. 10).

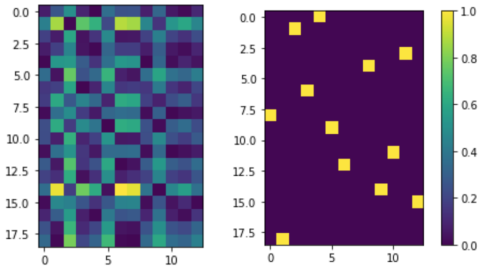
Fig. 10. 3D rays shot out from camera



The next step was to match players between views, to ensure we are mapping the same person from different views. Essentially, this would score how well each player in view 1 would match each player in view 2, with the score based on how closely the 3D keypoint rays intersect. An affinity matrix was made from these scores.

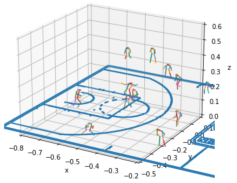
In order the find a matching of players in between views, the score matrix would need to become a permutation matrix. The nonzero elements would correspond to matches in players. The hungarian algorithm was used to find the permutation matrix that minimizes the cost of the affinity matrix. (Fig 11).

Fig. 11. Affinity matrix before and after being solved.



The correspondences were used to triangulate the players' key-points in 3D space and was displayed as seen in Fig. 12.

Fig. 12. 3D Triangulated Players



8. FUTURE WORK

8.1. Cycle Consistent Correspondence

We wish to use more than 2 views and have a cycle consistent player correspondence in between views [1].

8.2. Homography Between Frames

We could also use SIFT, matching, and RANSAC to find a homography matrix to take us from one frame to another. This homography could be decomposed to get changes in camera DoFs in between frames. This strategy is not prone to local minima in the loss function of the optimization.

8.3. Robust Smoothing over Frames

We could also optimize multiple frames camera DoFs at once while also adding a temporal loss to smooth out the camera movements.

8.4. Normal Estimations in 2D Point Cloud Matching

The loss function of the optimization could be improved to not pt2pt distance, but rather a pt2plane distance or correspondence matching that looks at normal vectors.

9. DIVISION OF WORK

Kush initially tuned and ran AlphaPose on our data, and then focused on the 2D ball pose estimation algorithm. Akarsh worked on camera projection, model processing, camera pose initialization and optimization, and 3D triangulation to generate the final output.

10. REFERENCES

[1] Junting Dong, Wen Jiang, Qixing Huang, Hujun Bao, and Xiaowei Zhou, "Fast and robust multi-person 3d pose estimation from multiple views," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019.

[2] Edoardo Remelli, Shangchen Han, Sina Honari, Pascal Fua, and Robert Wang, "Lightweight multi-view 3d pose estimation through camera-disentangled representation," 2020.

[3] M. Burenius, J. Sullivan, and S. Carlsson, "3d pictorial structures for multiple view articulated pose estimation," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, June 2013, pp. 3618–3625.

[4] Adrià Arbués-Sangüesa, Coloma Ballester, and Gloria Haro, "Single-camera basketball tracker through pose and semantic feature fusion," 2019.

[5] Thomas Haenselmann, Marcel Busse, Stephan Kopf, Thomas King, and Wolfgang Effelsberg, "Multicamera video-stitching," 01 2006.

[6] Donglai Xiang, Hanbyul Joo, and Yaser Sheikh, "Monocular total capture: Posing face, body, and hands in the wild," 2018.

[7] Hao-Shu Fang, Shuqin Xie, Yu-Wing Tai, and Cewu Lu, "RMPE: Regional multi-person pose estimation," in *ICCV*, 2017.

[8] Jiefeng Li, Can Wang, Hao Zhu, Yihuan Mao, Hao-Shu Fang, and Cewu Lu, "Crowdpose: Efficient crowded scenes pose estimation and a new benchmark," *arXiv preprint arXiv:1812.00324*, 2018.

[9] Yuliang Xiu, Jiefeng Li, Haoyu Wang, Yinghong Fang, and Cewu Lu, "Pose Flow: Efficient online pose tracking," in *BMVC*, 2018.