Robust Camera Pose Estimation and 3D Human Reconstruction for Sports Events

Jing Huang Hanrong Zhuang Lin Zhang Yuxiang Liu
Kun Li*
Tianjin University, China
{hj00, zhr_2021, zhanglin_just, lyx1021, lik}@tju.edu.cn

Abstract

This is a technical report for the FIFA Skeleton Light Challenge 2025. Given a video of a single broadcast camera, the task is to predict the 3D skeletons of players in the world coordinate system. We propose a method that extends our work, RCR (Robust Crowd Reconstruction), framework to video inputs. To estimate camera pose, we design a relative camera pose search algorithm with a fast line projector and achieve robustness and efficiency. To estimate 3D skeletons, we use the RCR framework to estimate the Human-scene Virtual Interaction Point (HVIP) and the SMPL parameters. Finally, we refine the 3D HVIP to ensure the consistency of the human movement and 3D skeletons are extracted from the SMPL parameters.

1. Method

Given a video, we first remove the distortion of each frame (Sec. 1.1) for a simplified projection model, then estimate the camera pose of each frame (Sec. 1.2) and estimate the camera space SMPL parameters of each frame (Sec. 1.3), which can be converted to the world space as the final output by the esitmed camera pose.

1.1. Data Preposessing

Undistortion. The precisely calibrated camera intrinsics of each frame are provided as input. For each frame, they are intrinsics matrix $K_{\text{original}} \in \mathbb{R}^{3\times3}$ and distortion coefficients $D_{\text{original}} \in \mathbb{R}^5$ (opencv format). For convenience, we undistort the image for a unified camera model, which is the camera intrinsics matrix $K \in \mathbb{R}^{3\times3}$ without distortion. Note that:

- Removing the distortion is necessary for our method.
- Keeping a unified K is not necessary. We make it unified only for convenience. Later, when we disccuss the speed of our method, we will make fair tests on the frames where distortion is removed but intrinsics matrixes varies from frame to frame. (Because a unified K needs upsampling the image to around 5K to make it lossless, resulting in slow processing speed.)

1.2. Camera Pose Estimation

In this subsection, the inputs are:

- A unified camera intrinsics matrix K.
- A undistorted image sequence $\{I_i\}_{i=1}^N$.
- A sparse man-made point cloud of the football court to define the world space.
- A camera extrinsics matrix $E_{\text{first}} \in \mathbb{R}^{4\times4}$ of the first frame, which is the transformation from the world space to the camera space.
- Players' ground-truth bounding boxes.

Our goal is to estimate the camera extrinsics matrix $E_i \in \mathbb{R}^{4\times4}$ of each frame I_i. Inspired by the official baseline[§], we solve the camera pose by finding the best alignment between (1) the detected football court lines and (2) the projected world space court lines.

2D Court Line Detection. We first detect a coarse line mask following the baseline via an adaptive thresholding method. Then we enhance it with the following strategies: (1) since we find our later steps are robust so that the previous frame's camera pose is with high confidence, we use it to remove the area that is outside the 2D projected football court (expand the 3D court by 3%); (2) we remove the given human bboxes area to make it look clean; (3) we remove the pixels that are not "white enough" (the euclidean distance between its color and $[0, 0, 1]$ is larger than 0.7 in HSV space); (4) we use the opencv `dilate` function with 2 iterations to make the line mask thicker.

3D Court Line Projection. We first convert the sparse point cloud to 3D vertices and their connections. This conversion needs a little manual work, but it is acceptable because we only need to do it once. Given a current i-th

*Team supervisor.

§Official baseline: https://github.com/G3P-Workshop/Skeletal-Tracking-Starter-Kit/

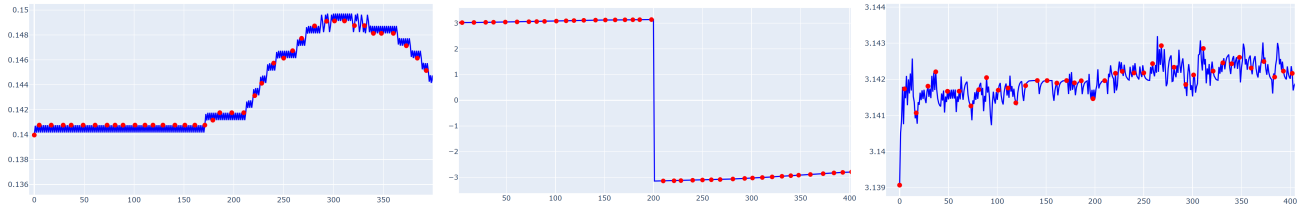

Figure 1. The camera poses sovled by our searching algorithm and line projector. The red points are those with the highest IoU scores among every 15 frames. This example video is ARG_CRO_000737.

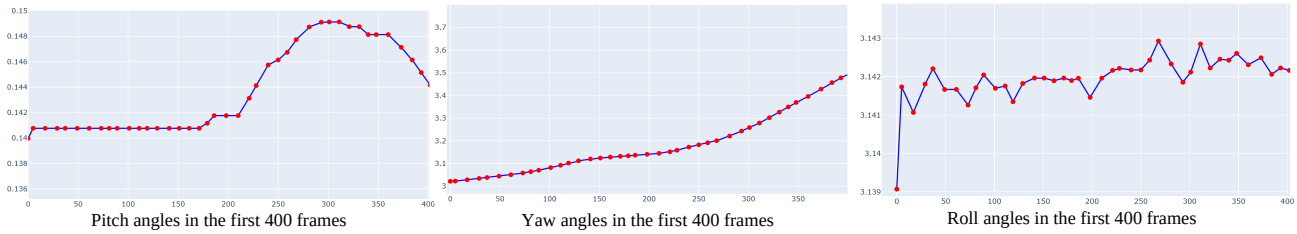

Figure 2. The camera poses refined by the red poses. The red points are those with the highest IoU scores among every 15 frames. The stretegy of refinement is to keep the red points only and interpolate others linearly. This example video is ARG_CRO_000737.

frame's camera extrinsics $E_i \in \mathbb{R}^{3\times4}$ and the intrinsics $K \in \mathbb{R}^{3\times3}$, we can first project the 3D vertices to 2D image space and draw white lines on a black image according to the connections. For the original 1080p resolution, we make the width of the lines 4 pixels. Finally, we convert the image to a mask in the Boolean type.

Camera Pose Searching. Inspired by the official baseline, we ignore the translation part of the camera pose and solve the camera rotation part only. We note that (1) the degrees of freedom of the camera pose is now only 3, which is a very small value, and (2) the previous 3D line projection process is fast enough (more than 100 FPS). We can afford to search the camera pose in a brute-force way. Therefore, we first convert the camera extrinsics E_i to 3 actual euler angles (yaw, pitch, roll) and then try a linear space search. In practice, for yaw and pitch we search in $[-0.385°, 0.385°]$ with 10 uniformly sampled steps, and for roll we search in $[-0.075°, 0.075°]$ with 8 uniformly sampled steps. For each combination of the 3 angles, we compute an IoU (Intersection over Union) score between the detected line mask and the projected line mask and select the one with the highest score. An example of our searching results is shown in Fig. 1. According to our test, this algorithm is not sensitive to the searching range, *e.g.*, $[-0.5°, 0.5°]$ with 7 steps can also work well for all the videos. We also discuss our choice between searching and differential rendering in the discussion section. Finally, we only keep the camera poses with the highest IoU scores among every 15 frames and interpolate the camera poses in between linearly to avoid the jittering. An example of the refined camera angles is shown in Fig. 2.

1.3. Camera Space SMPL Reconstruction via RCR

We use a recently proposed method, RCR(Robust Crowd Reconstruction)[2], to reconstruct SMPL parameters for each frame. RCR can take a single image as input and first estimate the camera and ground parameters, and detect all the individuals in the image. Since camera poses have been nicely estimated in the previous step, and tracking information is also given, for each frame, The inputs of RCR are (1) the frame image, (2) the 2D keypoints of each given bounding box, (3) the camera intrinsics K and (4) the ground parameters $G = \{N, D\}, N \in \mathbb{R}^3, D \in \mathbb{R}$, where 2D keypoints are obtained from RTMLib[1, 3] detecting the bounding boxes and G represents the ground plane consisting of those 3D points that satisfy the equation $N^T \cdot P + D = 0$, where P is a 3D point in the world space. G can be converted by the camera extrinsics E_i since the ground plane in the world space is $z = 0$. The RCR's outputs of each individual are the 3D SMPL pose parameters $\theta_{cam} \in \mathbb{R}^{72}$, shape parameters $\beta \in \mathbb{R}^{10}$, rotation $R_{\text{toCam}} \in \mathbb{R}^{3\times3}$, scale $S_{\text{toCam}} \in \mathbb{R}$ and translation $T_{\text{toCam}} \in \mathbb{R}^3$.

1.4. Post-processing

HVIP Smoothing. RCR models the position of the human by the torso center point and HVIP (Human-scene Virtual Interaction Point), where the torso center point is the average of the 4 key body joints (left shoulder, right shoulder, left hip and right hip) and the HVIP is a projected point of the torso center point on the ground plane. To avoid the jit-

tering of the human position, we smooth the HVIP by (1) removing the outliers, (2) interpolating the removed points linearly, and (3) smoothing the sequence. The above process will be processed in the world space (x, y only since $z = 0$ for all HVIPs) for several times. We illustrate the HVIP sequence after this post-processing in Fig. 3.

In the end, we convert each SMPL parameters to joints in the world space to satisfy the requirement of the challenge.

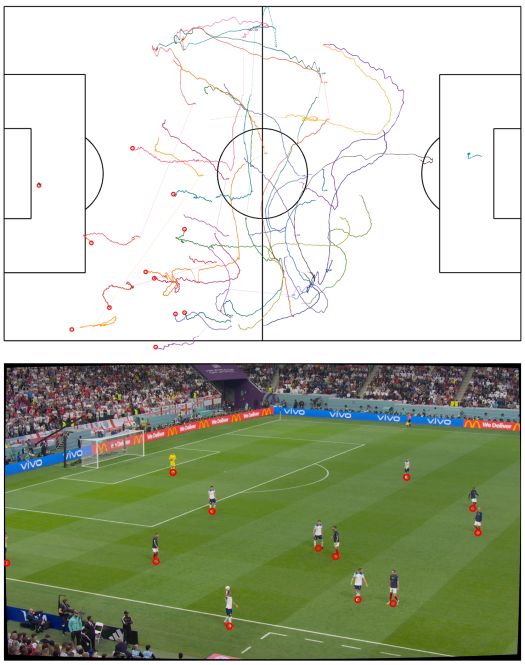

Figure 3. The upper image shows the smoothed HVIP trajectory in world space, projected onto the ground plane ($z = 0$, *i.e.*, the football court surface). The red circle marks the HVIP at the first frame. To illustrate this correspondence, the lower image shows the input image at the first frame, with the same red circle indicating the person's position.

2. Experiments

2.1. Datasets and Training

Datasets and Training. We set the combination of the training set of *LargeCrowd*[2] dataset and the first 80% objects of *worldpose* dataset to be the training set and the rest 20% objects to be the validation set. We train the HVIPNet of RCR[2] on the training set and use the pretrained model for other learnable modules. To balance the training speed and the ratio of the 2 sub-training sets, we randomly sample 10% of the *WorldPose* subset for every epoch. We train the model for 40 epochs and use the weights with the best validation loss for the final submission.

2.2. Results Visulization

We show the qualitative results of our method in Fig. 4.

Figure 4. A qualitative visualization of our results in both front view reprojection and a third view.

3. Disccussion

Differential Rendering vs. Searching with Line Projector Using a differential rendering framework (*e.g.*, Pytorch3D) with its silhouette renderer is a reasonable and elegant solution to the camera pose estimation problem. However, we find that the camera pose freedom can be considered to be very small in our challenge, and some of the team members are undergraduate students who are not familiar with rendering and optimization. Therefore, we choose a simple searching algorithm with a line projector to solve the camera pose. Consequently, it proves to be fast (all 13 video can be done in less than 2.5 hours in a server with an AMD EPYC 7543 CPU and no GPU) and robust (the algorithm is not sensitive to the searching ranges and steps). Note that we use the same hyperparameters for all steps and videos from start to finish.

References

[1] MMPose Contributors. Openmmlab pose estimation toolbox and benchmark. https://github.com/open-mmlab/mmpose, 2020. 2

[2] Jing Huang, Hao Wen, Tianyi Zhou, Haozhe Lin, Yu-Kun Lai, and Kun Li. RCR: Robust crowd reconstruction with upright space from a single large-scene image. *arXiv preprint arXiv:2411.06232*, 2025. 2, 3

[3] Tao Jiang, Peng Lu, Li Zhang, Ningsheng Ma, Rui Han, Chengqi Lyu, Yining Li, and Kai Chen. Rtmpose: Real-time multi-person pose estimation based on mmpose, 2023. 2