3D Human Pose Estimation

Phase II Report

Farhanur Rahim Ansari, Vidhey Oza, Minji Lee

Abstract

The project aims to perform 3D human pose estimation on monocular RGB images and videos and make an interactive tool that helps in using this technology with convenience. Throughout the semester, we dedicated phase I to scrutinizing well-known human pose estimation models and setting up a working prototype and pipeline. During Phase II, the inherent problems of human pose estimation models such as jittering, occlusion, and depth-ambiguity were fixed by improving the 2D Keypoints prediction step by shifting to the HRNet model from the OpenPose model implemented during the 1st phase. In the final pipeline, the HRNet model receives the input image or video and produces 2D output, then the GAST-Net model receives the 2D input and returns the 3D output. Finally, Django was used to build a web application, and the deployment was done on Amazon AWS Cloud Server for strong computational power.

Summary

Human Pose Estimation is an important problem and has enjoyed the attention of the Computer Vision community for the past few decades. It is an essential step towards understanding people in images and videos. Given an image of a person, 3D Human Pose Estimation (3D HPE) is the task of mapping the spatial position of a person into a simulated 3D space.

Human pose estimation is a well-solved problem in real-world scenarios for the 2D case. However, recovering 3D pose from 2D RGB images is considered more difficult than 2D pose estimation due to the larger 3D pose space and more ambiguities. An algorithm has to be invariant to factors like background scenes, lighting, clothing, skin color, and image imperfections, among others. 3D HPE has immediate applications in various tasks such as action understanding, surveillance, human-robot interaction, motion capture, and CGI. We can enjoy the fruits of 3D-HPE while sitting on a comfortable couch and watching Disney movies, or we can actively engage with 3D-HPE technology by playing against a virtual tennis player on Nintendo Wii or Xbox Kinect.

Having already developed a model working prototype pipeline for generating 3D Pose Estimations from input videos, our objective for Phase II was to further extend the development for handling in-the-wild inputs. The main goals of this phase were to fix existing identified problems in the pipeline, exploring new methods that can give us better results, and developing an interactive tool for using this technology. 3D HPE research is well tested in the indoor and lab settings, but not too much on in the wild dataset. Our exploration, which will be detailed out in the following sections, provided us insights into which direction we should march on.

As previously mentioned during phase I, our final goal was to make an interactive tool that can be used to perform the task of 3D HPE easily by anyone for fun and make improvements on the quality of its output. The crucial part of our goal is to go from proof-of-concept to real-world (in-the-wild) images and videos. This project will not only automate the tasks being done manually previously but also will help us gather interesting insights from the extracted information that was previously not possible.

Challenges to 3D HPE

3D HPE comes with a set of challenges, which makes it a more difficult problem to generalize for in-the-wild cases. Some of these challenges are inherent to the field of computer vision, while others are very specific to 3D HPE.

I. Self-Occlusion

Self-Occlusion is a complex problem in virtual environments, and it occurs when part of an object gets hidden from a camera view behind the object itself. For example, in the example below, the left hand and leg of the batsman are not visible to the camera and hence its pose estimation is particularly difficult to achieve.

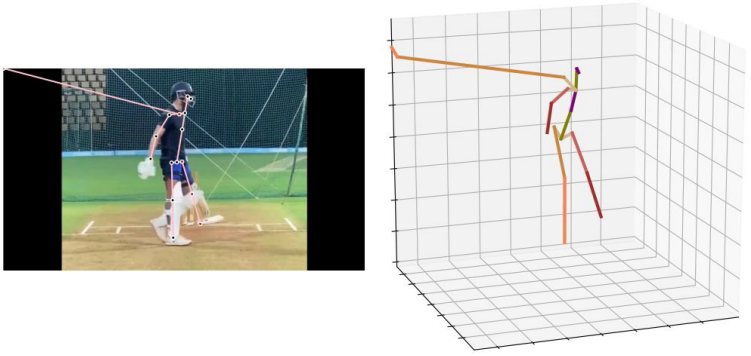

Fig. 1: (Phase I output) The problem of self-occlusion: the left hand of the batsman is not visible, and since the 2D pose estimation module does not detect and track it correctly, the error is propagated to the 3D projection module too

II. Depth Ambiguity

Depth in computer vision is estimated from the 2D images obtained through frame capture in the case of videos. From a 2D image, all the object seems to appear at an equal distance from the camera, which in reality may not necessarily be true. In such a case, estimating the depth at which each object needs to be placed while performing the 3D transformation is a challenging task. For example, in Fig. 8, the relative depth between the two legs is not clear from the 3D output.

III. Jitter

The problem of jitter is one of the most common ones faced in any computer vision application. It mainly occurs when the performed task in the video is a bit complicated, and the number of frames per sec in the video is not enough to capture the nuances of the actions. As a result of this, the input video is not a true representation of the actual task being performed. When the 3D transformation is applied to this video and if the 3D HPE model is not prepared to handle the jitter case perfectly, the final output appears to be a bit shakier than the ground truth. For example, in Fig 8, we can see that the spine of the batsman in the output image seems to be at a bit more tilted angle than the original image. And if this in continuous frames, then it appears to be a bit unstable.

Datasets

It is very challenging to build an in-the-wild dataset. Finalizing and training the model on the right dataset was one of the most crucial and challenging parts of 3D HPE. We spent a good amount of time exploring and deciding on them during Phase I. Let's quickly take an overview of the final datasets which we have decided on for the 2 different stages of our project.

For training our model to produce 2D keypoint predictions from the input video, we have worked on the COCO and MPII Human Pose dataset. Both of these are publicly available benchmark datasets for object detection, segmentation, and pose estimation. Each of them has 50K+ images with 80+ subjects covering 300+ human activities.

For transforming the estimated 2D keypoints into 3D pose estimation, the models were trained and tested on two state-of-the-art datasets: Human3.6M and HumanEva-I. Human3.6M has 3.6 million video frames with 11 professional subjects performing 15 daily activities. HumnaEva-I, on the other hand, is a much smaller dataset and contains only 4 subjects with 6 everyday actions.

Fig2: Sample Images from Human3.6M dataset

Methods

After doing a thorough research and analysis in phase I, we decided to move ahead with a two-step approach. The first step is the prediction of 2D keypoints from the input video, and the next step is 3D pose reconstruction from the 2D keypoints. Our implementation in Phase 1 used OpenPose as the first module of the pipeline since GAST-Net provided similar support to almost all 2D-HPE models, and OpenPose has been the popular choice for 2D pose estimation since their first release. However, OpenPose output was not fully compatible with GAST-Net input formats and required in-depth analysis and alteration of OpenPose code, which seemed too far-fetched given the project time period. So we decided to make alterations in the phase I pipeline.

I. Input-to-2D Pose Estimation – HRNet

HRNet is one of the state-of-the-art 2D pose estimation methods developed by [7] by designing an ingenious method inspired by the architecture of U-Net. It uses three different kinds of convolutional operations (as given in Fig. 2): 1-to-1 convolutions that maintained the size of the input image to output, high-to-low subnetworks that performed max-pooling to reduce the image size, and low-to-high subnetworks that performed upsampling and unspooling to increase the image size. These 3 connections were used together, alternating with a simple 1-to-1 connection so as to maintain and update information across all scales created by the convolution operations.

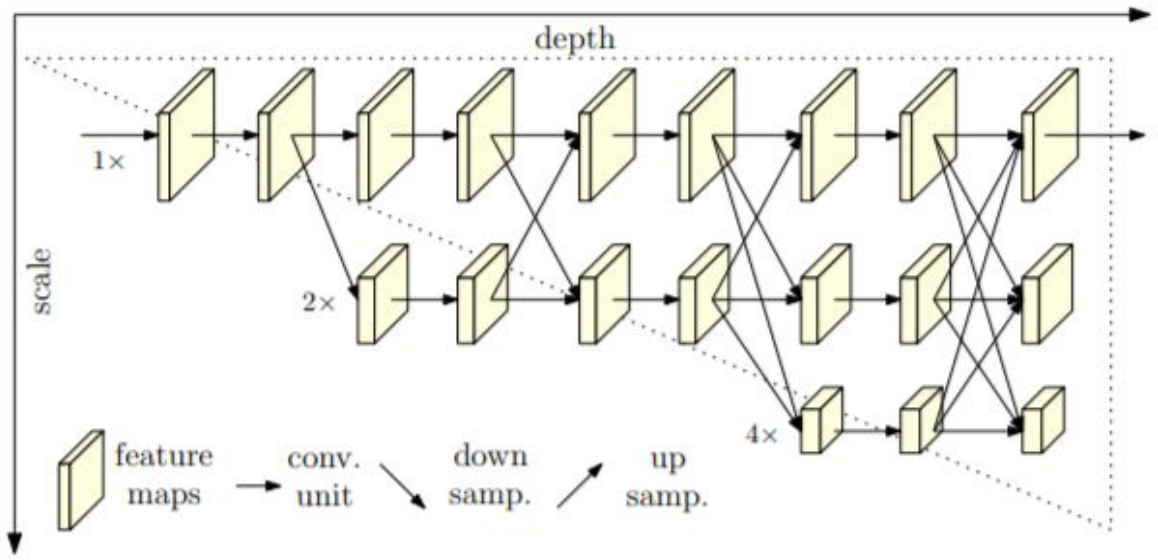

Fig. 3: Architectural design of HRNet (excerpt)

During Phase II, GAST-Net released extended support for HRNet, where they guided developers on how to integrate HRNet with GAST-Net since that was their own model of choice at the time of development. So we moved to HRNet after testing for potential problems like proper output formatting or GAST-Net compatibility. This was easy to do since we had already worked on multiple 2D-HPE models like OpenPose, Integral Pose, and HRNet.

II. 2D-to-3D Pose Estimation – GAST-Net

GAST-Net is the current state-of-the-art 3D HPE model that does not have its own 2D HPE auxiliary model developed and instead relies on the user to implement their own desired choice of the 2D module to suit needs depending on their respective use cases. Since Phase II did not involve significant changes in this section of the pipeline, we will only provide a brief summary of the model here in the Phase II report.

GAST-Net is an attention-based model that relies on three kinds of subnetworks to perform stable 3D pose estimation while tackling challenges like self-occlusion and depth ambiguity.

The spatial subnetworks are divided into two categories: the local attention and global attention graphs. Local attention graphs extract information from the adjacent keypoints for estimating the position of the given keypoint. For example, for a left elbow, the adjacent keypoints are the left shoulder, left wrist, and neck and face keypoints. On the other hand, the global attention graphs consider the given keypoint's relative positioning with all the keypoints for the body. In our implementation, that means all the 16 keypoints affect the position of the left elbow, however infinitesimal each contribution may be. The temporal subnetwork architecture makes use of the information captured in the time dimension and stabilizes the predictions made by the spatial attention network architecture that precedes it.

These spatial and temporal subnetworks are used alternatively to extract information from their predecessors and apply their own convolutions to achieve highly stabilized 3D pose output that closely mimics the actual human pose. The formidable aspect of this model is that this performance of the model relies on a single camera input and its 2D pose, meaning the model truly understands depth perception and handles problems like occlusion almost as naturally as humans do.

Interactive Tool

1. AWS

Like any other machine learning/deep learning models, HRNet and GAST-Net need high computing power. The digital videos or images they process could be as small as a few MBs to more than a few TBs. Thus, we chose to borrow the power of Cloud computing services; GCP and AWS are the top two choices.

Firstly, we explored GCP. With a free trial account, we're given $300 credits and 90 days of the free trial period. NVIDIA Tesla T4 model was suitable, which comes with a GPU and 13GB memory. Soon, we encountered problems in connecting SQLite, an embeddable relational database management system that comes with Django. SQLite did not recognize the path to which the video was uploaded and saved in GCP settings. Hence, we tweaked our plan and used AWS.

We uploaded Django projects on AWS instance. We chose ubuntu, 15.5GB memory, and CPU. All the video input formats are .mp4, and the output video is named 'animation_input name.mp4' and saved in the same directory as the input video. The issue regarding SQLite appearing at GCP settings was resolved easily in AWS settings.

2. Django

Django is a Python-based free and open-source web framework that follows the model-template-views(MTV) architectural pattern. The Model helps to handle databases. It is a data access layer that handles the data. The Template is a presentation layer that handles the User Interface part completely. The View is used to execute the business logic and interact with a model to carry data, and renders a template.

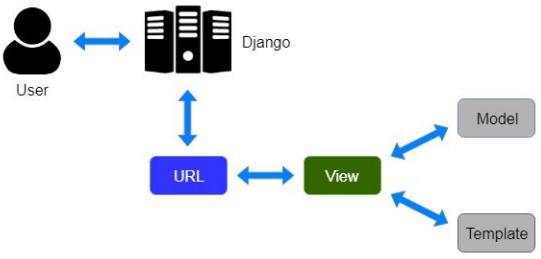

Fig. 4: MVT based control flow

We chose Django as our web framework as it is suited for large projects; it has built-in functions, a well-organized document, and an active community to help us navigate. The version we used is 3.1.4. We built two pages, index.html, and after_index.html. The home page (index.html) has a box where users can put in the caption for a video, and below is a button from which users can upload the video to run the models and get 3D output. Once users upload a video, it is directed to after_index.html.

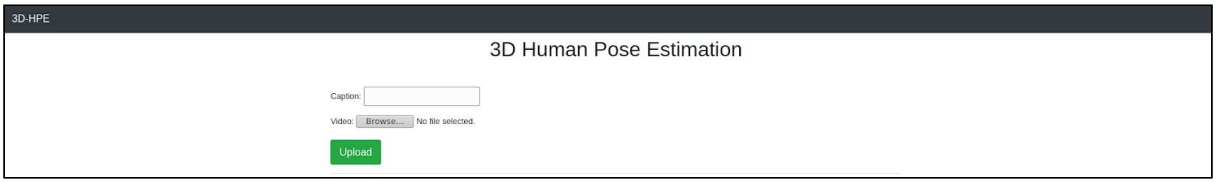

Fig 5: The snippet of the home page (index.html)

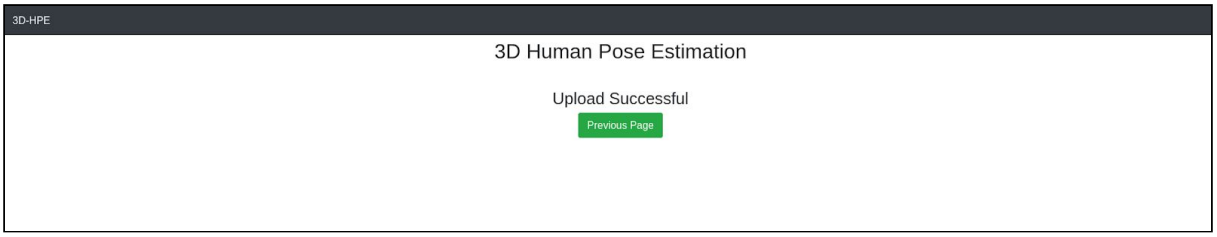

Fig 6: The snippet of the page after submitting a video (after_index.html)

Once it is done processing, users click the 'Previous' button, and then they will be redirected back to index.html, which is now showing the input video and the output video.

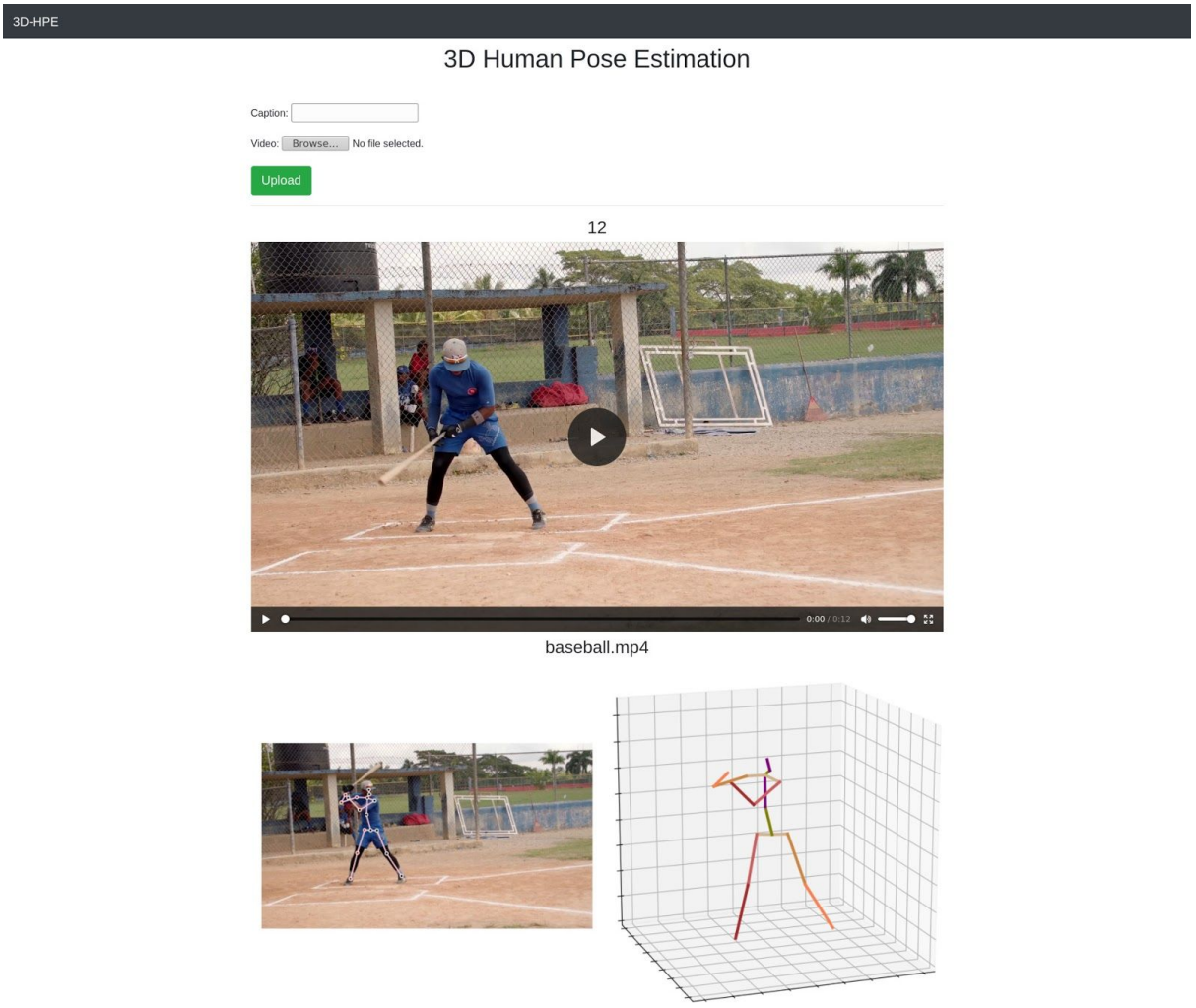

Fig 7: The input and output videos are shown on the index.html page

Results

First, we discuss the quantitative results of our pipeline modules. Then we will show how that performance translates to visual improvements while overcoming the common challenges of 3D HPE.

HRNet uses the mAP metric to tune and test its library of models against the existing industry standards. mAP stands for Mean Average Precision, which is the mean of all average precisions at a given similarity threshold. HRNet uses the COCO dataset for training, which uses Object Keypoint Similarity (OKS) where threshold value 0.5 has results provided by the most number of models. At a

high level, OKS is a Gaussian normalized distance between ground truth and predictions. Table 1 shows the benchmarking done against existing models like CPN, Integral Pose, and Mask-RCNN.

GAST-Net uses a slightly different metric called Mean Per Joint Positioning Error (MPJPE), which is similar in logic to mAP but has its own simpler mathematical foundation: it calculates the average error across all keypoints between the ground truth and predicted pose. Table 2 shows it's benchmarking against Pavllo, a popular industry standard that inspired many future model architectures for 3D pose estimation.

mAP	**COCO**	**MPII**
HRNet	91.5	92.3
CPN	74.9	77.0
Integral Pose	67.8	-
SimpleEnsemble	-	91.5

Table 1: Benchmarking with HRNet

MPJPE	**Human3.6M**	**HumanEva-I**
GAST-Net	23.11	21.2
Pavllo	34.5	35.2

Table 2: Benchmarking with GAST-Net

An interesting observation to make is that HRNet and GAST-Net both outperform the benchmarks (or are very close), but their real novelty is being able to achieve similar performance on random unseen data.

One of the biggest challenges that we overcame was the issue of self-occlusion and how it was originally handled by HRNet and GAST-Net. While GAST-Net had provisions to make sense out of occluded keypoints as opposed to out-of-frame keypoints, the 2D pose estimation models all had their own way of handling keypoints invisible on the image. While OpenPose used a confidence score to show that a keypoint on origin combined with low confidence meant that the keypoint was not visible, HRNet simply removed that keypoint from its output matrix. As a result, we had to design a custom "bridge" between the 2D and 3D models in a way that translated what occluded keypoints meant in their respective standards.

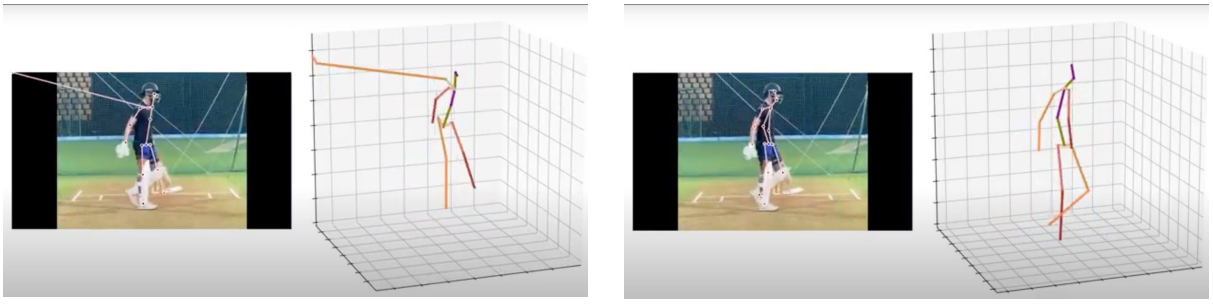

Fig. 8: Left image – the left hand is almost fully occluded behind the body of the batsman, causing issues when trying to estimate the position of what is not visible to the model at all. Right image – GAST-Net is able to infer the left-hand position when the 2D model is appropriately bridged to the 3D module. Also, note the slight bend in hand, showing its prowess in solving depth ambiguity as well.

Jitter is another such challenge that is inherent with any kind of pose estimation, and since we are performing 2D HPE followed by 3D HPE, we run the risk of amplifying the jitter from the first

module to the second. For this reason, the choice of the 2D model and its retraining was crucial because the lower jitter it had, the less it would propagate to the final output. For this reason, OpenPose and HRNet remained our best contenders, with HRNet being the final choice due to its easy trainability through its intuitive code structure, a better pre-existing library of model weights, and a speed similar to OpenPose. So with an HRNet + GAST-Net pipeline, we alleviated the problem of visual jitter in the final 3D pose output to a great extent.

Discussion

In this project, we aimed to solve the problem of 3D Human Pose Estimation on monocular RBG images and videos, and in the end, develop an interactive tool for using this technology with ease. We aimed to develop a computer vision application that can be applied for in-the-wild inputs. During phase I, we looked into pre-existing models and made working prototypes using the OpenPose model for 2D keypoints prediction and the GAST-Net model for 3D Pose Estimation. Our prototype performed decently well on simpler inputs but did not do that well on complicated ones. We faced the problems of jitter, depth ambiguity, and self-occlusion, which were inherent to the domain.

During phase II, our main goal was to fix the existing identified problems, improve the scalability of our model, and develop a web application. For fixing the existing issues, we tried fine-tuning and retraining the pipeline but didn't achieve significant results. We explored different architectures and got the best result by replacing OpenPose with HRNet in the pipeline. GAST-Net also has better support for HRNet, which makes it a better fit in the pipeline. We then fine-tuned and retrained this new pipeline, which helped us in making the model more scalable and simultaneously fixing the pre-existing issues.

For the development of an interactive tool, we explored and did thorough research on all the available options for a python based web development application and its deployment on a cloud server. We developed the application using Django and deployed it on the AWS cloud server. We initially started with GCP but had to shift to AWS midway because of the technical difficulties in setting up the SQL instance on GCP. We also used AWS for retraining our model.

Overall, we were able to achieve the goals set for phase II and for the entire semester-long project. This project has given us a better understanding of the computer vision domain. Though there were few roadblocks in the journey, we were able to successfully solve those problems. In doing so, we have developed skill sets, both technical and otherwise, which are required for working in a real-world data science project right from its conceptualization to the finished product.

Statement of Contributions

- Farhanur Rahim Ansari – Django, GCP, Scalability
- Vidhey Oza – HRNet, Fixing issues, Scalability
- Minji Lee – Interactive tool, Django, AWS, GCP

Github Link

https://github.com/mjlee2121/3D-HumanPoseEstimation

References

1. *Raj, B. (2019, May 01). An Overview of Human Pose Estimation with Deep Learning. Retrieved October 01, 2020, from Medium.*
2. *Mehta, D., Rhodin, H., Casas, D., Fua, P., Sotnychenko, O., Xu, W., & Theobalt, C. (2017). Monocular 3D Human Pose Estimation in the Wild Using Improved CNN Supervision. 2017 International Conference on 3D Vision (3DV), 506-516.*
3. *Hidalgo, G., Raaj, Y., Idrees, H., Xiang, D., Joo, H., Šimon, T., & Sheikh, Y. (2019). Single-Network Whole-Body Pose Estimation. 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 6981-6990.*
4. *Liu, J., Guang, Y., & Rojas, J. (2020). GAST-Net: Graph Attention Spatio-temporal Convolutional Networks for 3D Human Pose Estimation in Video. arXiv preprint arXiv:2003.14179.*
5. *Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh, "OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields," in IEEE Transactions on Pattern Analysis and Machine Intelligence, doi: 10.1109/TPAMI.2019.2929257.*
6. *Pavllo, D., Feichtenhofer, C., Grangier, D., & Auli, M. (2019). 3d human pose estimation in video with temporal convolutions and semi-supervised training. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 7753-7762).*
7. *Sun, K., Xiao, B., Liu, D., & Wang, J. (2019). Deep high-resolution representation learning for human pose estimation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 5693-5703).*