Human Pose Estimation for 3D Point Clouds
CS221 Final Project Report

Matthew Daly
madaly16@stanford.edu

1 Introduction

Human pose estimation is a very important topic in the field of computer vision. Postures and poses contain a lot of information about the activities taking place in a scene, as well as the emotions and social cues present. Therefore, estimating human poses from images is an important step to understanding scenes with people.

Pose estimation from 2D images has had a lot of success with the advent of deep learning, and especially with convolutional neural networks (CNNs). State of the art methods use popular 2D CNN architectures that are successful in image classification, and train them to regress the locations of pose keypoints from images of people.

3D human pose estimation takes in a 3D image and produces the 3D (x,y,z) coordinates of the keypoints. 3D images can be produced from stereo cameras and LIDAR scanners, and capture information about depth in the image. This provides a lot more information about the objects in the image, and allows a deeper understanding of the scene the image depicts. This is especially true with 3D images of people, as the depth component provides meaningful information about how people are posed in 3D space.

I propose a method of using deep learning to create a model that can estimate human poses from 3D images.

2 Literature Review

2.1 2D Pose Estimation

As mentioned before, 2D pose estimation has had a lot of success with deep learning methods that train CNNs to regress keypoints from images of people. These methods are even successful with estimating poses in images with multiple people, with notable approaches being DeepCut[1] and OpenPose[2].

2.2 Deep Learning on 3D images

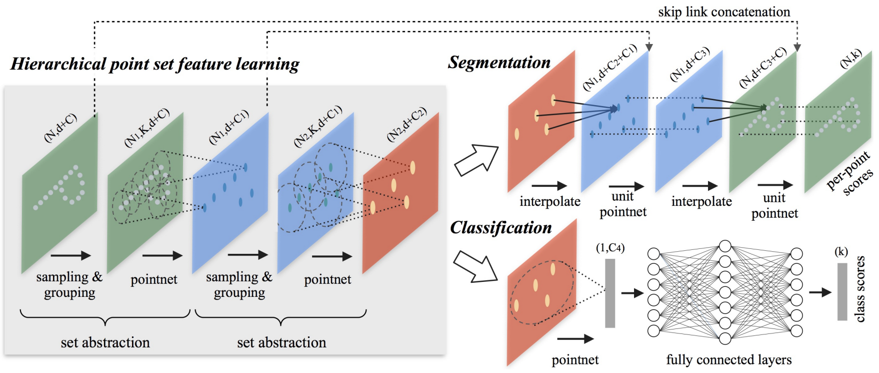
3D images have become more popular in computer vision due to their increased usefulness in applications such as autonomous driving and robot navigation. Likewise, 3D image data has become more abundant through the prevalence of 3D sensors like LIDAR sensors and stereo cameras, like Microsoft Kinect and Intel Realsense. This has naturally led to an interest in utilizing deep learning with this 3D data, though this has not reached the same level of success deep learning on 2D data has.

One approach has been to convert 3D images to 3D grids of voxels (the 3D equivalent of pixels) and apply 3D CNNs to the voxel grids. One example of this is VoxNet[3]. This approach has several drawbacks. One is that training and running 3D CNNs can be very computationally intensive. Another is that voxelizing a 3D image will likely lead to a large number of voxels representing empty space and essentially no information. Yet another drawback is that voxelizing an image can lead to quantization artifacts that prevent fine grained detail from being represented.

Another approach would be to work with point clouds, which are unordered sets of points in space that represent 3D objects and scenes. Point clouds are a sparse representation that can represent fine detail, so working directly with point clouds can be very beneficial compared to voxelizing the image.

A seminal paper for using deep learning directly on point clouds is PointNet[4]. PointNet uses a series of multi-layer perceptrons (MLPs) to learn per-point features, then aggregates all per-point features into a global feature vector using a max pool operation. This global feature vector can then be used for tasks such as classification, which PoinNet achieved state of the art results with. Aggregating the features with max pool is key because it allows permutation invariance. Permutation invariance is important when it comes to point clouds, because any permutation of the same set of points still represents the same image.

One issue with PointNet is the lack of learning local structures, which is important to understanding geometric structures of point clouds. To rectify this, PointNet++[5] was proposed, which stacked PointNet modules in a hierarchical fashion to learn more of the local structure of the point cloud. PointNet++ does this by sampling neighborhoods of points, based on the point clouds metric distance, and applying PointNets to these neighborhoods, then repeating for several layers. Please refer to the following diagram of PointNet++'s architecture.



Another important contribution to deep learning on point clouds is the Dynamic Graph CNN (DGCNN)[6] and its proposed EdgeConv operation. The goal of the EdgeConv operation is an improved method of learning local geometric features in point clouds and mimic the success the convolution operation has had in 2D images. EdgeConv works by first creating a k-nearest-neighbor (KNN) graph from the point cloud where each point is a node and it is connected by edges to its k nearest neigbors. For each point in the graph/point cloud, EdgeConv uses a shared weight MLP on the pont's edges to learn features for that point. The KNN graph is computed for each EdgeConv operation, so a point's nearest neighbors change each layer based on the feature space. EdgeConv has similar benefits to the convolution operation, where later layers are working with more abstract features and have a larger receptive field in the original point cloud. EdgeConv is also permutation invariant, which is an important property when working with point clouds. This useful EdgeConv operation can be placed in a variety of network architectures, and the DGCNN authors used it within a PointNet-like architecture to achieve state-of-the-art results in point cloud classification and segmentation.

To capture information about local structure in point clouds, approaches like PointNet++ and DGCNN must structure the point cloud data first. For example, PointNet++ uses farthest-point sampling to sample local subsets of points, and DGCNN creates KNN graphs to capture the geometric structure of the point cloud. This data structuring can be costly, and Grid-GCN[7] seeks to provide a more efficient data structuring method that "blends the the advantages of volumetric models and point-based models, to achieve efficient data structuring and efficient computation at the same time"[7]. Grid-GCN achieves this data structuring by creating a grid in the point space and assigning points to voxels in the grid. Then, points are sampled from the grid (in a way that only voxels containing points are considered), and a local graph is created where the sampled points are connected to their k nearest neighbors. A MLP is then used to learn features for the center point using the local graph, similar to the EdgeConv operation. With this approach, Grid-GCN achieved state-of-the-art results and time efficiency.

2.3 Deep Learning Approaches to 3D Pose Estimation

The general strategy for deep learning approaches to 3D pose estimation has been similar to 2D pose estimation approaches, which is to take deep learning approaches that have been successful in learning from the images and performing tasks such as classification, and to use those approaches to perform the task of regressing pose keypoints in the image.

V2V-PoseNet[8] utilizes the voxelization approach to create a 3D grid and uses a 3D CNN to regress the keypoints.

Point Based Pose Estimation (PBPE)[9] uses the standard PointNet as a base architecture and trains it to regress pose keypoints.

Zhou et al.[10] proposed an architecture using EdgeConv layers to regress keypoints with high accuracy.

3 Dataset

There is a lack of large, high-quality datasets for 3D pose estimation, unlike 2D pose estimation which benefits from benchmark datasets such as COCO and MPII. First of all, 3D sensors are no where close to ubiquitous like cameras, so 3D images of people are much less common. Secondly, labeling body keypoints in 3D is much more involved than annotating poses in 2D images.

ITOP[11] is the de facto benchmark for 3D pose estimation, with most methods reporting results on it. ITOP contains 100K 3D images of real people, with each image labeled with the 3D coordinates of 15 different body parts. The dataset consists of 20 different people performing 15 action sequences each, captured with a stereo camera.

Synthetic datasets are also reasonably popular for 3D pose estimation. They are created by generating point clouds from 3D models of people in different poses. The UBC3V[12] dataset is a notable example of a synthetic 3D pose dataset. It contains 300k samples of 16 different characters in different poses.

I will use ITOP to train and evaluate my model, but I may also utilize UBC3V for further training to improve performance.

4 Evaluation Metric

The standard metrics for evaluation 3D pose estimation are average precision (AP) and mean average precision (mAP). The convention in this field is to define AP for each body part keypoint as the percentage of predicted coordinates withing 10 cm of the ground-truth coordinates. Then, the mAP is the mean across the body parts of the AP.

5 Baseline

3D pose estimation has several defined baselines with results that have been reproduced independently. These baselines, usually using more classical machine learning approaches, are often referenced in 3D pose estimation papers to compare against the proposed implementation. A popular baseline that I will also use is the random forest approach[13] to estimating 3D poses. This has been shown to get 65% mAP on the ITOP dataset.

I also proposed and implemented my own baseline for this task. 3D images can be represented as 2D images where instead RGB channels or intensity channels, there is a depth channel representing the distance from camera. I created a 2D CNN model from the ResNet architecture with its pre-trained ImageNet weights, and trained it on the ITOP dataset to regress 3D keypoints. My initial results were very poor, with <1% mAP, but after fixing a bug in the evaluation code, utilizing normalization, and training for longer, my 2D CNN baseline achieved 20% mAP, which is still quite poor.

6 Main Approach

My approach is similar to other successful 3D pose estimation approaches, which is to take deep learning methods that have been shown to work well with 3D data in general, and apply it to pose estimation specifically. I propose utilizing the brand new Grid-GCN method, which has demonstrated state-of-the-art results on classification and segmentation tasks, and using it to regress 3D poses.

Adapting a point cloud classification network to regress 3D poses is as simple as modifying the final layers to return the desired output size, and then training on point clouds of people labeled with their pose keypoints. Though the authors of Grid-GCN provided a GitHub repository with their code, I decided to re-implement the network myself in MATLAB. This served multiple purposes. First, MATLAB is good for working with point clouds, and has built-in methods for visualizing and manipulating them. Second, re-implementing from scratch is a good exercise that solidifies understanding, while sometimes (including this case) being easier than trying to understand and undocumented code base. Finally, multiple implementations in different frameworks can propagate and foster understanding within the community.

Unfortunately, due to limited time and resources, I was unable to finish training my model and obtain meaningful results. However, I have laid important groundwork by solidifying my understanding of Grid-GCN, implementing a running model and training loop, and forming a plan to move forward. I plan to continue with this project, making improvements and successfully training a model and obtaining results.

7 Code

The code is publicly available on GitHub at the following link:

`https://github.com/mrdaly/point-cloud-pose-estimation`

The project is entirely implemented in MATLAB and uses MATLAB's Deep Learning Toolbox. MATLAB's PointNet documentation example (link) was drawn from to create the MLPs and custom training loop, and the Grid-GCN paper was used to implement the network's architecture and modules. The Grid-GCN GitHub (link) was mainly just used to obtain configuration parameters, such voxel grid sizes and MLP channel sizes.

8 Future Work

The current state of the project is a trainable model and training script that is able to successfully run through training examples. However, clearly there is much more work to be done. The first step would be to properly run through training of the dataset to obtain a fully trained model, then evaluating it to get results. Next, some key features of the network are missing from my implementation:

- Batch Normalization
- "Coverage aware sampling" of voxels instead of randomly sampling voxels
- Thresholding the point cloud and denoising to obtain the points of just the human with no background data

- Using ”semantic” relationships between points as features rather than just geometric features
- Data augmentation on point clouds to improve generalization
- Training on another dataset e.g. UBC3V

I hope to continue working on this project as I believe it will provide a meaningful contribution to the fields of 3D pose estimation and deep learning on point clouds.

References

[1] Leonid Pishchulin, Eldar Insafutdinov, Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, Peter Gehler, and Bernt Schiele. Deepcut: Joint subset partition and labeling for multi person pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[2] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. In *arXiv preprint arXiv:1812.08008*, 2018.

[3] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Ieee/rsj International Conference on Intelligent Robots and Systems*, pages 922–928, 2015.

[4] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv:1612.00593*, 2016.

[5] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.

[6] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 2019.

[7] Qiangeng Xu, Xudong Sun, Cho-Ying Wu, Panqu Wang, and Ulrich Neumann. Grid-gcn for fast and scalable point cloud learning. 2019.

[8] Gyeongsik Moon, Juyong Chang, and Kyoung Mu Lee. V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[9] Ashar Ali. 3d human pose estimation. Master's thesis, Georgia Institute of Technology, 2019.

[10] Yufan Zhou, Haiwei Dong, and Abdulmotaleb El Saddik. Learning to estimate 3d human pose from point cloud. *IEEE Sensors Journal*, 2020.

[11] Albert Haque, Boya Peng, Zelun Luo, Alexandre Alahi, Serena Yeung, and Li Fei-Fei. Towards viewpoint invariant 3d human pose estimation. In *European Conference on Computer Vision*, October 2016.

[12] Alireza Shafaei and James J. Little. Real-time human motion capture with multiple depth cameras. In *Proceedings of the 13th Conference on Computer and Robot Vision*. Canadian Image Processing and Pattern Recognition Society (CIPPRS), 2016.

[13] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Commun. ACM*, 56(1):116–124, January 2013.