CIS581: Final Project Report
3D Scene Flow for Mesh-Subject Correspondence Tracking

Aadit Patel
aaditp@seas.upenn.edu

Nicholas Gurnard
ngurnard@seas.upenn.edu

Rithwik Udayagiri
rithwiku@seas.upenn.edu

Ankit Billa
ankitb@seas.upenn.edu

Evan Grant
edgrant@seas.upenn.edu

Daniel Stekol
dstekol@seas.upenn.edu

Google Drive containing all submission files.

1 Abstract(Project Summary)

This project focuses on the applications of feature correspondence, homography estimation, human pose detection and mesh transformations. Given a video of an individual moving, a rigid 3D mesh is to be warped to the surface of a specified body part which moves with the body throughout the video. This movement takes into account the 2D movement across the image plane as well as the rotation about any axis that occurs in the video. For example, if the desired goal is to give a human Iron Man's helmet, then the head must be tracked in order to warp the helmet onto the head with the correct pose. The 3D pose of the object must be maintained within the 2D video, hence 3D scene flow. Traditional methods tend to struggle with accommodating changes to the 3D orientation of the subject, something we plan to explore in this work.

2 Goal & Objective

This project is split into 2 main goals/objectives -

1. Successfully track the 2D movement of the subject in image plane. This will involve human pose detection and feature point extraction from the subject, both of which are being taken care of using Google's MediaPipe framework [6].
2. Stick the external 3D object mesh onto a specified area of interest on the subject. We will compute the transformation matrix of the subjects 3D movements between 2 consecutive frames and using it to perform a rigid body transformation on the external object.

3 Related Work

- **Human Pose Estimation:** The authors of [3] released and demonstrated a novel method for real-time 2D multi-person pose estimation - and open-source library called OpenPose. This method takes a 2D RGB image input and outputs the 2D locations of anatomical keypoints of human figures in the image. This is done using a multi-stage CNN to first predict a set of confidence maps for body part locations, then generate part affinity fields which indicate the degree of association between body parts which comprise a single pose.

- **Rigid 3D Scene Flow:** The method presented by [4] involves a deep architecture capable of reasoning at an object level to describe a dynamic 3D environment using generalized optical flow. This strategy separates background (static) elements from dynamic foreground rigid body agents, and relaxes the supervision requirements for dense scene flow as a result of the object-level abstraction.
- **MediaPipe:** 3D human pose estimation with feature tracking for face and hands has been implemented in [6].
- **Optical Expansion:** The authors of [10] present a technique for using dense optical expansion - a cue of depth given by expansion of an image feature in the 2D image frame - for upgrading 2D optical flow to 3D scene flow.
- **3D SIFT Descriptor:** The authors of paper [8] propose a 3D SIFT Descriptor that can be used for action detection
- **Fourier Features to learn High Frequency functions**: [9] talks about using a Multi Layer Perceptron to learn high frequency parts after passing features through fourier functions.

4 Proposed Method

Our proposed approach can be divided into 5 sub-tasks: Human Pose Detection for Feature point extraction, load and render .OBJ file (3D object), scale and align .OBJ file to match first frame, calculate rotation and translation between frames, transform 3D object, and render each frame with 3D object superimposed.

5 Experimentation and Evaluation

5.1 OpenPose

The OpenPose library [3] was the initial candidate for the first step of the project: human pose detection. The openpose library used a pretrained deep learning model that could return a skeleton of a human in a video. However, OpenPose came with some huge disadvantages. The installation process was extremely difficult because of dependency issues that were not well documented, the computation time was very slow, and the output was a stick figure where the bones would be instead of a point cloud. After about a week of unsuccessful use of OpenPose, the team switched to the MediaPipe framework by Google which addressed all of these concerns.

5.2 MediaPipe

Similar to OpenPose, MediaPipe [6] is a deep learning network that returns human pose, but it also returns the pose in the form of a 3 dimensional point cloud in real time. The model is lightweight enough to be run on mobile devices. MediaPipe eliminated the need to custom generate a 3D mesh for the subject. The network has models that generate 3D point clouds for the face, hands, body, hair segmentation, iris tracking, and more in case the project is expanded. Using MediaPipe and OpenCV, a 3D point cloud for the face was generated for every frame of a video.

5.2.1 Pose Model

The initial model to generate the 3D point cloud for the face was the MediaPipe Pose Estimation model [6]. The output of this model gives an estimation of 33 landmarks across the entire body, 11 of which being on the face. The origin on the model is centered at the hips of the person in the video. Thus, when the human was at an orientation where their body was sideways relative to the camera frame (side profile), the model attempted to estimate depth of the occluded body parts such as the hidden arm, leg, hip, etc. In addition, if the human in the video was close enough to the camera to the point where most of their body was out of frame, then the model struggled to estimate where the hip origin was. Because of this, a lot of the face landmarks were poorly estimated in depth, and often in X and Y coordinates as well as shown below.

Figure 1: Original Orientation of Face

Figure 2: Tilted Orientation of Face

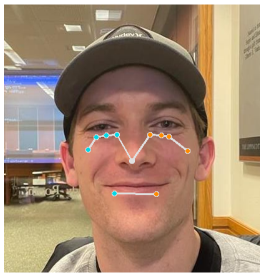
Figure 3: Pose Model Landmarks corresponding to Figure 1

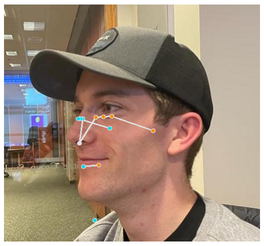
Figure 4: Pose Model Landmarks corresponding to Figure 2

Because of the poor estimations, the homography computations for rotation and translation were poor.

5.2.2 FaceMesh Model

The FaceMesh model in MediaPipe [6] was a much better estimator for 3D facial landmarks. The FaceMesh model generated a total of 468 different landmarks, therefore reducing the impact of an outlier landmark when computing the homography parameters.

When parts of the face are occluded, for example in Figure 6, the Face Mesh Model still estimated facial landmarks in 3D. The estimates proved to be much more accurate than those of the Pose Model, and thus generated much more meaningful homography parameters. The facial landmarks also happened to be ordered, meaning the tip of the nose landmark was the same position in the output array in Figure 5 as Figure 6. This detail eliminated the need to compute feature correspondence,

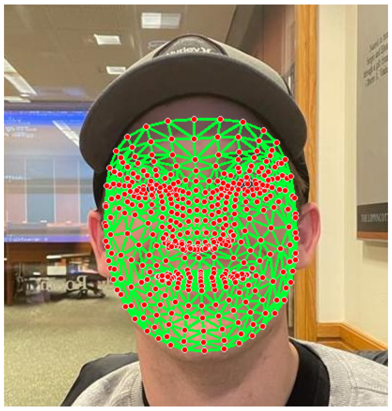

Figure 5: FaceMesh Landmarks corresponding to Figure 1

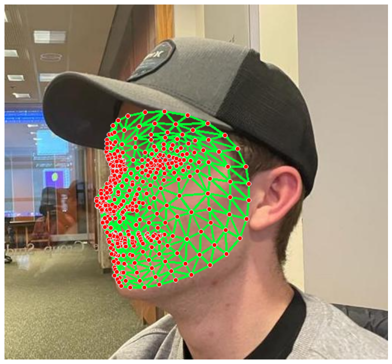

Figure 6: FaceMesh Landmarks corresponding to Figure 2

thus eliminating additional noise that may have been added to the homography parameters, and therefore the overall system.

5.3 Homography

Since the point clouds are in 3D space, ”homography” may be the abused in the sense that it is a 2D transformation. However, since the 3D point cloud was being projected into 2D space, the term homography was used to keep referencing this technique simple.

Given 2 3D point clouds, the homography parameters, being the estimated rotation matrix R and translation vector t, were estimated using the Kabsch algorithm [5] and least-squares minimization fitting [2]. Additionally, the face of the human was assumed to be a rigid body, thus simplifying the computation [1]. Assuming frame i has landmarks A and frame i+1 has landmarks B, the rotation matrix R and tranlation vector t are computed as follows:

$$RA + t = B$$

The first step to estimating the homography parameters was to compute the centroid of the 3D mesh vertices of 2 sequential frames. The centroid is just a mean of all of the points' (N points) coordinates, calculated as follows, where A is the matrix of points coordinates:

$$centroid_A = \frac{1}{N} * \sum_{k=1}^{N} A^k$$

Once the centroids are computed between frame 1 and frame 2, the rotation matrix R is estimated by aligning the centroids and using least squares to get the closest estimate [2, 5]. This eliminated the need to compute translation in the same step, simplifying the calculation. Singular value decomposition (SVD) could then be used on the covariance matrix H to obtain matrices U and V whose columns are the left-singular and right-singular vectors of the singular values in the diagonal matrix S.

$$H = (A - centroid_A) * (B - centroid_B)^T$$

$$[U, S, V] = SVD(H)$$
$$R = V * U^T$$

The translation between the 2 3D point clouds can then be computed using:

$$RA + t = B$$

$$t = B - RA$$

$$t = centroid_B - R * centroid_A$$

5.4 Pytorch3D Open3D Rendering

The .STL or .OBJ file is warped and rendered using PyTorch3D [7]. PyTorch3D is a tool used to load .OBJ files and store them as a mesh. It has multiple options for rasterizing, shading and rendering the mesh to an image. This library was used to transform the .OBJ file using the homography rotation matrix calculated to track the face of the person. Once the mask was successfully rendered, mesh vertices were selected using Open3D. These points selected on the Open3D mesh were then mapped to the corresponding location on the human face. Pytorch3D was then used again to superimpose the rendered mask onto the human face. The correct translation and scaling were applied in order for the helmet to look realistic when superimposed on the human.

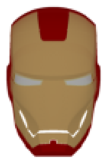
Figure 7: Mask render corresponding to Figure 1 before homography

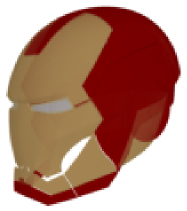
Figure 8: Mask render corresponding to Figure 2 after applied homography parameters

6 Results and discussion

The final result photos and video links are given in the appendix below. We were successfully able to extract the various points of interest from a human face, find the homography between the consecutive frames and apply that homography to a subject 3D mesh (in this case, an Iron Man helmet). We were also able to align the face and the 3D mesh correctly for every frame with appropriate scaling.

7 Future Work

The superimposed helmet was implemented in 2D, resulting in some instances where an ear or a nose would stick out of the helmet. A more robust superimposition can be implemented in 3D, such that the helmet perfectly wraps around the head. Additionally, objects such as gloves and body suit could be also be warped onto the human model instead of only a helmet.

References

[1] Finding optimal rotation and translation between corresponding 3d points. http://nghiaho.com/?page_id=671. Accessed: 2020-11-16.

[2] K Somani Arun, Thomas S Huang, and Steven D Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on pattern analysis and machine intelligence*, (5):698–700, 1987.

[3] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: realtime multi-person 2d pose estimation using part affinity fields. *IEEE transactions on pattern analysis and machine intelligence*, 43(1):172–186, 2019.

[4] Zan Gojcic, Or Litany, Andreas Wieser, Leonidas J Guibas, and Tolga Birdal. Weakly supervised learning of rigid 3d scene flow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5692–5703, 2021.

[5] Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976.

[6] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, Wan-Teh Chang, Wei Hua, Manfred Georg, and Matthias Grundmann. Mediapipe: A framework for building perception pipelines, 2019.

[7] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[8] Paul Scovanner, Saad Ali, and Mubarak Shah. A 3-dimensional sift descriptor and its application to action recognition. In *Proceedings of the 15th ACM International Conference on Multimedia*, MM '07, page 357–360, New York, NY, USA, 2007. Association for Computing Machinery.

[9] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020.

[10] Gengshan Yang and Deva Ramanan. Upgrading optical flow to 3d scene flow through optical expansion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1334–1343, 2020.

8 Appendix

The link to google drive can be found here. This drive contains the presentation, presentation video, output videos and final code used in this project. Some of our results are mentioned below.

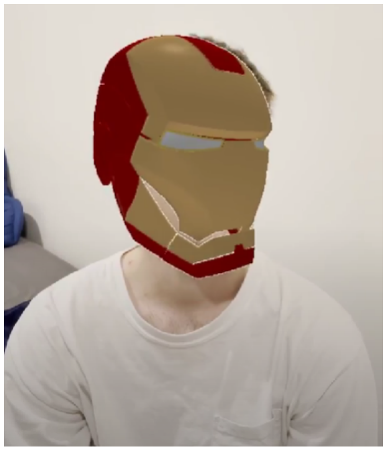

Figure 9: Superimposed output showing rotation

Figure 10: Original image

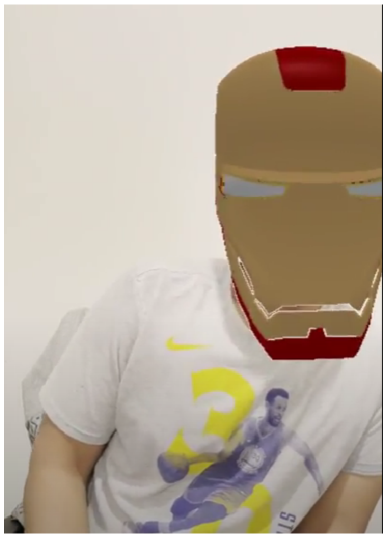

Figure 11: Superimposed output showing translation

Figure 12: Original Image