Addition of Accessories on 3D Face Model

Roshan Naik
Computer Science and Engineering
Indian Institute of Technology Delhi
New Delhi, India
jroshannaik@gmail.com

Parth Singh
Computer Science and Engineering
Indian Institute of Technology Delhi
New Delhi, India
parthsingh97@gmail.com

Prem Kalra
Computer Science and Engineering
Indian Institute of Technology Delhi
New Delhi, India
pkalra@cse.iitd.ac.in

***Abstract*—Most of the work on adding accessories onto faces has been done in 2D. These methods use deep learning models and directly process the 2D images. In this paper, we perform this process in 3D space. This is done by converting 2D images into 3D face models and then placing the 3D model of the accessory on this face model. We have limited the accessory to spectacles. The automation has been done by using feature points and Procustes analysis. We have also performed the similar automation in Blender using the python API. The fitting of several glasses and their adjustment with varying faces is also done through automatic scaling, skeletal rigging and bending of glasses.**

***Index Terms*—3D alignment, facial accessories, mesh stitching**

I. Introduction

Addition of accessories such as glasses, goggles is a very popular computer vision and graphics problem. A lot of private companies have developed their versions of solving this problem. Such as Lenskart Try On Glasses that uses the Ditto Technology and Glasses.com

The method used in these technologies is partly Machine Learning based wherein a generic face model of the face is constructed from an input of face images. The method used by Ditto [1] takes a video of the person turning his head from left to right, and then constructs a face model by detecting certain key positions of the face, such as points on the ear, eyes, nose and forehead. It then places a 3D model of the glasses onto the face mesh and then renders the frames with varying head positions with the glasses added.

The framework for our process involves a pipeline of various steps. The first step is the generation of the Morphable face model from the input image. There are plenty of techniques available for generation of 3D face models from a single input face image. For the purposes of demonstration, we have used Scalismo Morphable Model [2] to generate the 3D face. Once we have the generic face, we have to map the colors from the input image to the 3D mesh so that the colors look similar.

In the next phase, we attach a generic head to the constructed face. This process requires stitching and although there are methods to carry this out, we have developed our technique to stitch the face to the head. This step is necessary for the alignment of the glasses that comes next and also for future purposes as it gives an idea about the location of ears, forehead, etc.

In the last phase, we add the 3D model of the glasses onto the 3D model of the complete face, and render the scene with all sorts of reflection & refraction properties present on the glasses. We have two ways to perform this step. The first makes use of Procrustes scaling and alignment, the second uses Blender API & tools for the automated alignment.

The paper is structured as follows : Section II describes the theory behind Morphable Models and how we used it in our pipeline. Section III discusses stitching algorithm that we used for stitching the the face to the generic head. Section IV presents the alignment procedure that we used for automatically fitting the glasses onto the face. Section V presents other accessories that were experimented with and demonstrates the working of the pipeline through various examples and results.

A. Related Work

Our paper builds upon the technologies and techniques developed by various authors in a lot of past research in Graphics and Vision such as 3D modelling, Morphable modeling, realistic rendering, 3D registration, stitching, alignment, etc.

The first phase of our pipeline works by constructing a 3D model of a face from a single 2D image [6]–[8]. This is a very popular problem and is a subclass of 3D reconstruction problem [2]–[5]. Some of these papers use more than just a single RGB image. We however are dealing with the problem of 3D model generation from a single 2D image, and within this problem, we are specifically targeting to reconstruct a face of a human. This problem has been solved in the literature by the help of Morphable Face Models. Machine Learning, Deep Learning techniques have been quite popular in solving this via Morphable Models [7], [9].

In recently published work, [11] Risubh Parihar's paper deals with the same problem of adding accessories on the face, however, his paper focuses more on applying makeup and relighting the 3D face model, and less on adding accessories like glasses or hat, which are preformed only in 2D. We borrow the first step of his pipeline for the generation of 3D face from 2D image [1], [12] but develop novel techniques of our own to complete addition of 3D accessories to a fully 3D reconstructed head.

II. 3D Face Model

A. Morphable Model

To generate a 3D frontal face model from a given 2D image, we have used scalismo-faces. This tool uses Basel Face Models as a database of Gaussian Process Morphable

Models. It uses Analysis by Synthesis approach to find the best model corresponding to the input image. This approach first synthesizes the model and then checks the likelihood of the model with the input image. It follows a Markov Chain Monte Carlo simulation to fit the Gaussian Process (i.e. the Morphable Models are modelled as a Gaussian Process) to the input image data [1].

In order to generate the model, several parameters such as the shape of the Morphable Face Model (i.e. components of distortions that come from PCA of the population of distortions from the Basel Dataset), illumination, pose, distance from the camera, etc. are estimated and then the likelihood of the projected image is computed based on its similarity with the input image, then a slight perturbation to the parameters according to the algorithm of Markov Chain Monte Carlo simulation and the process is repeated until the iteration count is met. The larger the iterations, the better is the fit that we obtain if the model is converging to the face. Sometimes however, the model doesn't converge which happens mostly because of a very bad initialization of the model.

All the final parameters from this approach is stored in the Render Parameter File. These Render Parameters are needed to synthesize an image of the Morphable Model and compare it with the input image.

B. 2D Texture Mapping

Once the model is generated, we save it as a 3D mesh in '.ply' (Stanford Polygon Format) file. Now we need to map the texture of the original image onto this '.ply' mesh. The '.ply' file uses vertex colour. Also, since there is a correspondence between the rendered image of the Morphable Model and the input image, we simply replace the RGB values of the Morphable Model with the RGB values of the pixel corresponding to that vertex of that Morphable Model.

Fig. 1: Scalismo generated model

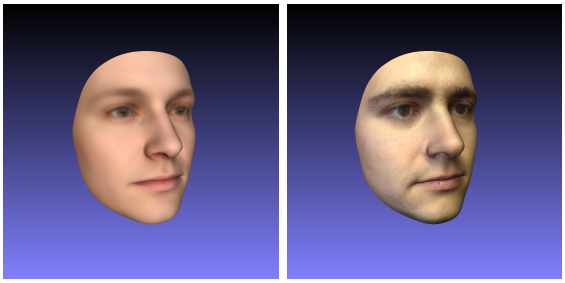


Fig. 2: Texture mapping

III. Head Stitching

For the next step of the pipeline, we have to extend the face into a complete head. This is done by stitching a generic head model that is available in 'ply' format with the face model that is generated in part II - 3D Face Model.

Stitching of meshes has been an entire subject of it's own in other papers such as in [13], [14]. The paper on Joint Alignment and Stitching of Non-Overlapping Meshes (JASNOM) [13] defines stitching as the process of joining two meshes say $M_1 = (V_1, E_1, F_1)$ & $M_2 = (V_2, E_2, F_2)$ which are 2-manifold meshes (i.e. have a $\mathbb{R}^2$ topology) with a boundary/hole. In this process a mesh $M_c = (V_s1 \cup V_s2, E_{stitch}, F_{stitch})$ is generated such that the resulting mesh $M = M_1 \cup M_2 \cup M_3$ is also a 2-manifold with $\mathbb{R}^2$ topology but without any boundary.

The connecting mesh M_c stitches the boundary of mesh M_1 with boundary of mesh M_2 seamlessly keeping the topology preserved. Since the connecting mesh doesn't introduce any new vertices it just includes V_{s1} which are the boundary vertices of M_1 and V_{s2} which are the boundary vertices of M_2 and connects all boundary vertices from one mesh to the other mesh. Also, each edge in E_{stitch} joins one vertex from V_{s1} to one vertex from V_{s2} and the edges don't cross over one another. There shouldn't be an edge that connects an interior vertex from any of the meshes, since then it would break the $\mathbb{R}^2$ topology of that mesh.

Although there is a developed algorithm [13] for automatically aligning two meshes appropriately and stitching them into a complete mesh, we have developed our own stitching algorithm for this special case of stitching a face mesh Fig. 2 with the back-head model Fig. 3 that obeys the constraints and specifications previously defined.

The stitching by our algorithm has the following steps. We first select a 3D model of a complete generic head, that includes the ears, forehead, scalp & face. We then carve out the face portion meticulously using a 3D modeling software like Blender. The resulting model is called the back-head. Refer on Fig. 3.

The next stage is to extend the face mesh so that it smoothly merges with the back-head. For this, we extract the boundary points of the face and the back-head. We then iterate over each boundary point in the face mesh and extend it to the closest boundary point on the back-head contour, call it a matching point. For those points on the back-head contour

which don't have a corresponding point that they are matched with on the face mesh boundary, an inverse matching point on the face mesh boundary is found. New edges and faces are then created such that all the vertices from both the boundaries are connected in an appropriate nearest-neighbor fashion and the resulting mesh has no boundary vertices. This is saved as a 'ply' file.

The final stage after creation of the new ply file is to polish it. This is done by performing some operations in Blender. In the edit mode, we perform "Remove Doubles" followed by "Recalculate Normals". Then we select the "smooth" shading instead of the "flat" shading in the Tools section for the purposes of better rendering in the software and Fig. 5 shows the results.

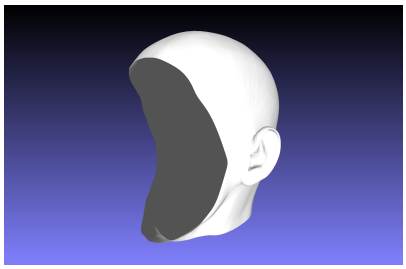
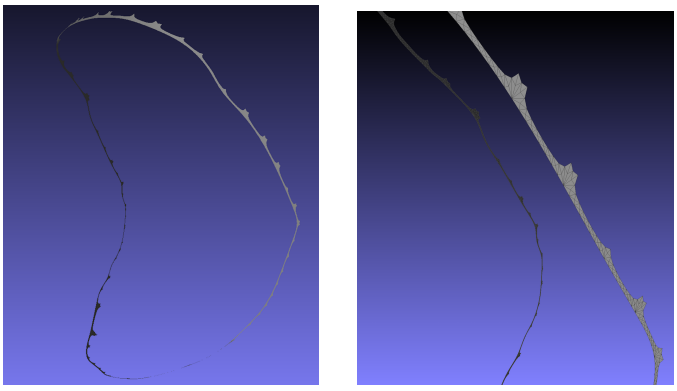
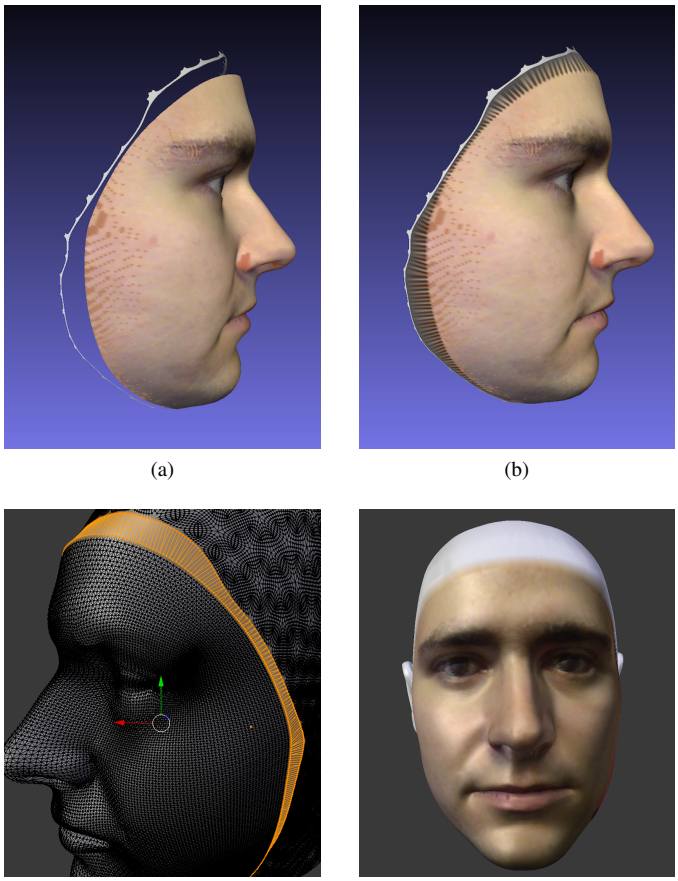


Fig. 3: The back-head model with the face carved out



(a) (b)

Fig. 4: Back-Head Boundary Contour



(a) (b) (c) (d)

Fig. 5: Stitching Results: (a)-(b) Side view of the face and back-head boundary before & after stitching, (c) Interconnecting mesh, (d) Final Render

IV. Alignment of Glasses

A. *Feature Points*

Once we have the face mesh and the mesh of the glasses to be placed, we choose certain feature points. These feature points are chosen so that there is a one-one correspondence matching between the feature points of the face mesh and glasses mesh. We have used the points on the side temple and bridge of the nose as feature points. Similarly, we use the points on the arms and nose support for the feature points on the glasses.

B. *Procrustes Analysis*

To align the glasses and the face mesh, we use the Procrustes analysis algorithm [10]. The algorithm takes as input, two point clouds. These point clouds, in our case, are the points clouds composed of the feature points. The algorithm tries to find the best transformation matrix such that distance between the points clouds is minimized. There are many ways we can calculate the distance between the meshes. One way is to sum the square of distances between each feature point and finally take the square root. This is referred to as Procrustes Distance.

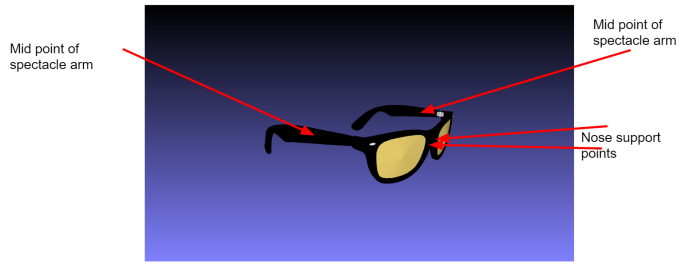


Fig. 6: Feature points of glass model

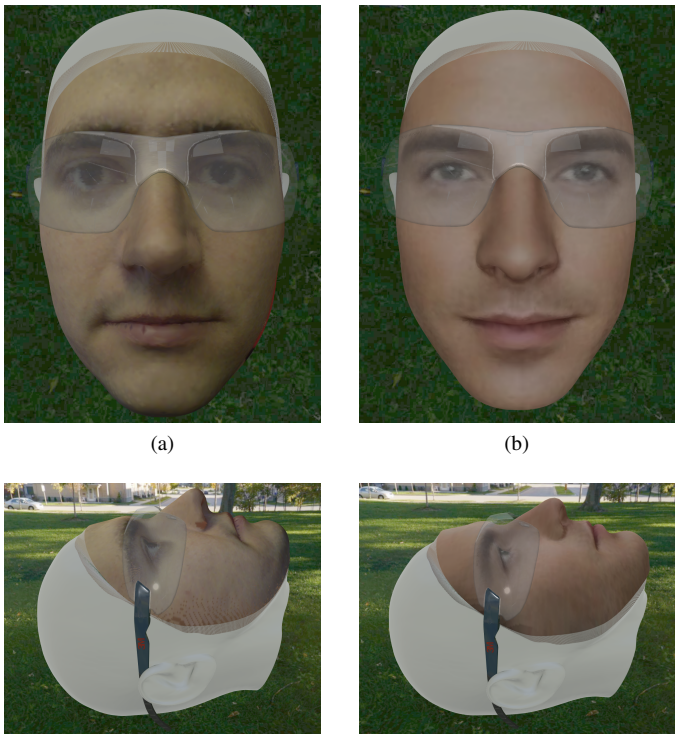
$$d = \sqrt{\sum_{k=1}^{n} (x_k - u_k)^2 + (y_k - v_k)^2} \quad (1)$$

Where, (x_k, y_k), k = 1 to n are the points of the first cloud and (u_k, v_k), k = 1 to n are the points of the second cloud.

The algorithm transforms the point clouds into a normalized space. The algorithm works in 3 steps:

1) Translation is performed so as to bring the mean of both point clouds at the same point. For the normalized space that point is the origin.
2) Once the mean of the point clouds becomes the same, scaling is performed on the point clouds so that the variance of both the clouds becomes equal.
3) The algorithm then tries to minimize the angle between each pair of feature points by rotating the point clouds.

Fig. 7: Glass aligned with the face model



(a) (b) (c) (d)

Fig. 8: Blender Alignment Results: (a)-(b) Front view (c)-(d) Side view

V. Accessories and Results

The key advantage of having a 3D model of the face and stitching it to a complete head is that various accessories can be added to it in 3D and it can rendered onto a 2D screen realistically. We have experimented with introducing a hat and spectacles.

Fig. 9: Hat aligned with the face model

We have also coloured the back head of the model to match the colour of the skin of the face. This is achieved by a machine learning algorithm called K-Means Clustering. It is a

clustering algorithm that clumps all the pixel colours found on the face into a certain fixed number of clusters and the colour of the cluster with the highest frequency is used to colour up the whole back head. Since, it is the most dominant colour on the face, it looks harmonious on the head.

The alignment of the hat was carried out manually, however automating can again be done using the same procedure described in Section IV but applied to a hat. Fig. 9 is rendered with 'Blender Render' renderer on Blender. The alignment of the spectacles through various techniques has already been explained in the previous sections and will be left out from this section.

REFERENCES

[1] Marcel Luethi, Thomas Gerig, Christoph Jud and Thomas Vetter, "Gaussian Process Morphable Models" IEEE Transactions on pattern analysis and machine intelligence, VOL. 40, NO. 8, August 2018.

[2] B. Goldluecke and M. Magnor, "Joint 3D Reconstruction and Background Separation in Multiple Views using Graph Cuts," Madison, Wisconsin, USA, IEEE Computer Society, Vol. I, 683-694, June 2003.

[3] B. Goldluecke and M. Magnor, "Space-Time Isosurface Evolution for Temporally Coherent 3D Reconstruction," Washington, D.C., USA, IEEE Computer Society, Vol. I, 350-355, July 2004.

[4] B. Goldluecke and D. Cremers, "A Superresolution Framework for High-Accuracy Multiview Reconstruction," Jena, Germany, 2009.

[5] Haefner, B., Peng, S., Verma, A., Queau, Y., Cremers and D., "Photometric Depth Super-Resolution," *Submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) Special Issue on RGB-D Vision: Methods and Applications*, 2018.

[6] Zou C., Liu J., Liu J. (2013) Precise 3D Reconstruction from a Single Image. In: Lee K.M., Matsushita Y., Rehg J.M., Hu Z. (eds) Computer Vision ACCV 2012. ACCV 2012. Lecture Notes in Computer Science, vol 7727. Springer, Berlin, Heidelberg

[7] Jackson, Aaron S, Bulat, Adrian, Argyriou, Vasileios & Tzimiropoulos, Georgios, "Large Pose 3D Face Reconstruction from a Single Image via Direct Volumetric CNN Regression," International Conference on Computer Vision, 2017.

[8] Diego Rother and Guillermo Sapiro, "3D Reconstruction from a Single Image," IEEE Transactions on pattern analysis and machine intelligence.

[9] Anh Tuan Tran, Tal Hassner, Iacopo Masi and Gerard G. Medioni, "Regressing Robust and Discriminative 3D Morphable Models with a very Deep Neural Network", 2016.

[10] J.C. Gower, "Generalized procrustes analysis", Psychometrika, 1975.

[11] Rishubh Parihar, Aniket Dashpute and Prem Kalra, "Scene Adaptive Cosmetic Makeup Transfer," ICVGIP, 2018.

[12] Sandro Schnborn, Bernhard Egger, Andreas Morel-Forster and Thomas Vetter, "Markov Chain Monte Carlo for Automated Face Image Analysis," International Journal of Computer Vision 123(2), 160-183 , June 2017

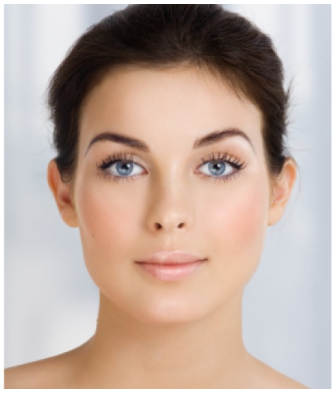
[13] Brando, Susana & Costeira, Joao & Veloso, Manuela. "Effortless Scanning of 3D Object Models by Boundary Aligning and Stitching." VISAPP 2014.

[14] Turk, G., Levoy, M., "Zippered polygon meshes from range images." In: SIGGRAPH 94, New York, NY, USA, ACM.

Input Image

(a)

(b)



(c)

Generated 3D Head

(d)

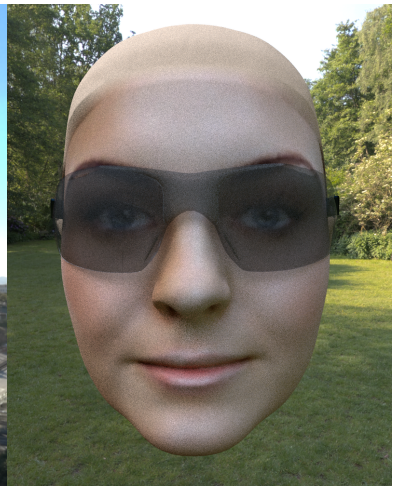
(e)

(f)

Accessories

(g)

(h)



(i)

Fig. 10: Top to Bottom: Pipeline of the process demonstrated using three examples

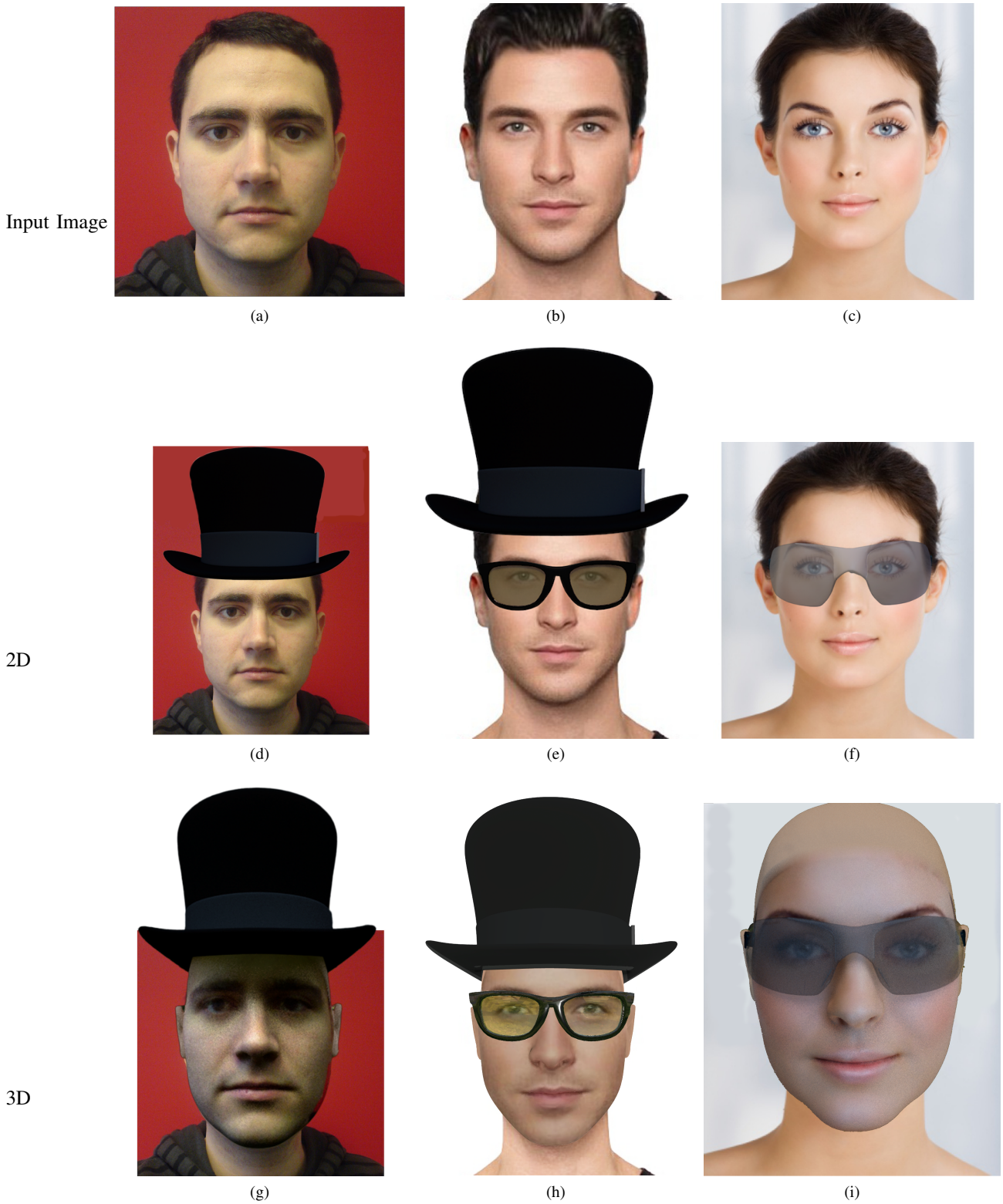


Fig. 11: Comparison of 3D accessories addition to 2D accessories addition. The 2D version in second row is just pasting of 2D projections of the 3D counterpart object in the third row. The 3D version displays realistic shadows, reflections and other light properties, many of which are not possible in the 2D version.