applied sciences

MDPI

Article

A Framework for Real-Time 3D Freeform Manipulation of Facial Video

Jungsik Park [1], Byung-Kuk Seo [2] and Jong-Il Park [1,*]

[1] Department of Computer Science, Hanyang University, 222 Wangsimni-ro, Seongdong-gu, Seoul 04763, Korea; nangsik@mr.hanyang.ac.kr

[2] Electronics and Telecommunications Research Institute, 218 Gajeong-ro, Yuseong-gu, Daejeon 34129, Korea; byungkuk.seo@etri.re.kr

* Correspondence: jipark@hanyang.ac.kr; Tel.: +82-2-2220-0368

Received: 20 September 2019; Accepted: 31 October 2019; Published: 4 November 2019

Abstract: This paper proposes a framework that allows 3D freeform manipulation of a face in live video. Unlike existing approaches, the proposed framework provides natural 3D manipulation of a face without background distortion and interactive face editing by a user's input, which leads to freeform manipulation without any limitation of range or shape. To achieve these features, a 3D morphable face model is fitted to a face region in a video frame and is deformed by the user's input. The video frame is then mapped as a texture to the deformed model, and the model is rendered on the video frame. Because of the high computational cost, parallelization and acceleration schemes are also adopted for real-time performance. Performance evaluation and comparison results show that the proposed framework is promising for 3D face editing in live video.

Keywords: 3D image/video editing; face editing/manipulation; augmented reality; parallel processing

1. Introduction

Social media has played a significant role on the way people can communicate and interact with others. With the advances in mobile devices and camera technology, activities such as taking and sharing photos or videos are increasingly a part of daily life. In particular, the "selfie" has become a dominant feature of online social media such as Facebook, Snapchat, or Instagram. This trend has been further boosted by filtering and editing capabilities, for example, making a face more beautiful or funny.

As a new form of entertainment on social media, face editing (or manipulation) in photos or videos has been demonstrated by a variety of commercial tools such as the well-known professional software, Photoshop or mobile applications such as FaceApp [1], Face Warp [2], Face Warp 2 [3], Plastic Surgery Simulator [4], and Snow [5]. However, such solutions have a number of drawbacks from both a practical and technical point of view. For instance, Photoshop provides delicate functionalities for editing photos of human faces, but it takes time and effort for users to become familiar with these capabilities. FaceApp allows users to change facial appearance more easily, but fully automated operations often lead to undesired results. Face Warp provides some useful editing functions, but only for frontal faces. With Plastic Surgery Simulator, users can freely manipulate both facial and body photos, but it is difficult to make delicate or fine changes. Face Warp 2 and Snow have the advantage of being able to easily edit both photos and live videos of human faces, but the editable parts and range are limited to a few templates. Furthermore, most such face editing applications are based on 2D image processing, and so incur background distortion. For example, manipulating the face in Figure 1a using the existing applications can result in background distortion, as shown in the red circles of Figure 1c,d. To address these performance limitations, we propose a new framework that allows 3D

freeform manipulation of a face in live video. Unlike existing approaches, the proposed framework fully employs 3D geometry of a face and provides improved results with real-time performance in video frames.

Figure 1. Results of face manipulation: (**a**) original image; (**b**) proposed framework; and comparison with existing applications: (**c**) Plastic Surgery Simulator; and (**d**) Snow. Red circles indicate image distortions incurred by manipulations.

Techniques for 3D editing of image and video have been diversely researched in the field of computer graphics [6–11]. The main flow commonly involves:

1. acquiring a 3D model of a target;
2. registering the 3D model and a 2D image of the target;
3. manipulating the model using 3D geometric transformation or deformation; and

4. texturing the manipulated model and rendering it on the image.

Several approaches have been used to acquire a 3D model of a target, including computer vision-based 3D reconstruction from multiple images [6,7], interactive 3D modeling from a single image under the assumption that its 3D shape is symmetric [8,9], and obtaining from online 3D model database [10,11]. Manipulation has also been implemented in various ways including copy and paste, geometric transformation, and isotropic or anisotropic scaling [8–11]; deformation to an arbitrary shape [7,8]; and physically based deformation [6,11]. Even though such methods allow image or video editing based on 3D geometry of the target, they are unsuitable for face manipulation because the human face's range of expressive deformations cannot be modeled by a rigid body or a simple physical models as in the above methods.

Face manipulation has been extensively studied and demonstrated on many different features such as transferring facial expressions to another face [12], replacing a face with another face [13], reshaping specific parts of a face [14,15], and facial animation for a digital avatar [16–19]. In some details relevant to our work, Kitanovski et al. [14] projected a sparse 3D face model on a 2D face image before and after deformation and computed the affine transformation using the corresponding triangles of the projected 2D meshes. This approach allowed natural and efficient manipulation without any visible distortions in real time with the help of a GPU. Since the image warping based manipulation can suffer from arbitrary rotation of head, they demonstrated it as augmented reality mirror, an appropriate application that does not require large head rotations. Similarly, Cao et al. [15] showed fine manipulation results, but the proposed method could not be applied in live video. Moreover, as deformation was achieved by adjusting the weights of predefined attributes mapped to a bilinear face model, the deformable range was highly dependent on the bases of the face model. Nagano et al. [19] implemented facial animation for a digital avatar that could be instantly generated even from a single image and rendered to reflect lighting conditions as well as expressive details (e.g., wrinkles and inside of mouth). However, deformation was confined to the range of human facial expression—in other words, it was difficult to manipulate faces into arbitrary shapes that were funny or exaggerated faces.

Our framework can overcome these limitations as well. In the proposed framework, a 3D face model is registered on a face region in a video frame by detecting facial landmarks [20] and then fitted to a 3D morphable model (3DMM) [21]. At the same time, the 3D model is deformed by a user's input based on the as-rigid-as-possible (ARAP) method [22]. Finally, the video frame is mapped to the deformed model as a texture, and the model is rendered on the video frame. By taking account of the face's 3D geometry, the proposed framework supports natural 3D manipulation of a face without background distortion (Figure 1b) and enables visualization of the deformed faces from different viewpoints across video frames. As faces can be interactively manipulated by the user's input, this allows freeform manipulation without any range or shape limitations. Building on our previous work [23], this study presents a complete framework with real-time performance based on parallelization and acceleration schemes, which is much sought after for practical applications. This study also details performance evaluation and comparison with existing applications.

2. Facial Manipulation Framework

As shown in Figure 2, the proposed framework consists of three modules: registration, manipulation, and rendering. In the registration module, facial landmarks are detected from the video frame, and states of the 3D face model (position, orientation, shape, and expression) are estimated. At the same time, in the manipulation module, the mean face model in the neutral state is deformed by the user's input, and the deformation is applied to the current state of the face model. Finally, in the rendering module, the deformed model of the current state is rendered on the video frame.

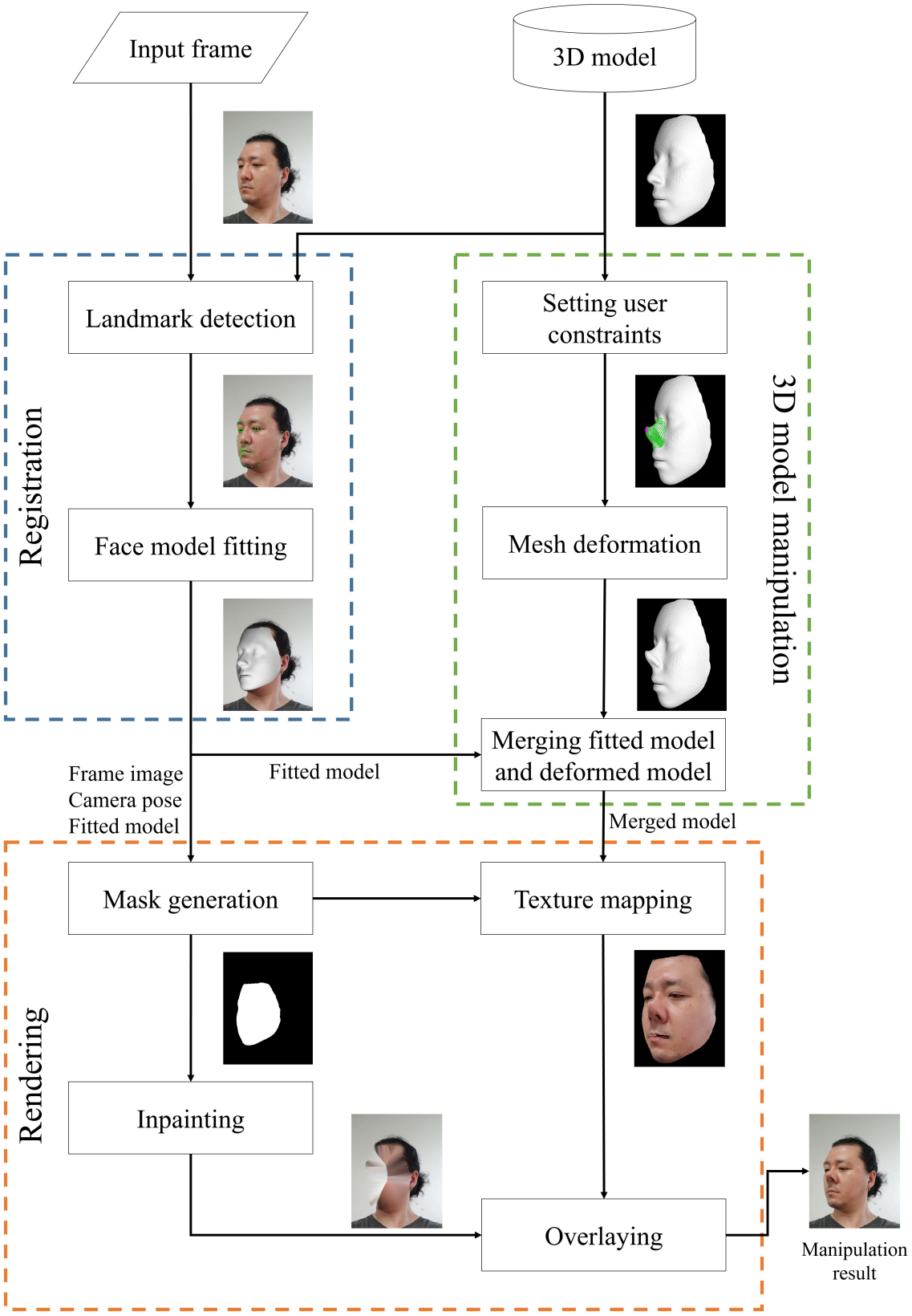


Figure 2. Workflow of the proposed face manipulation framework.

2.1. Registration

The registration process involves two steps: facial landmark detection and face model fitting. For facial landmark detection, feature points (or interest points) around the eyes, nose, mouth,

and facial outline are detected in a video frame. These points are then used as reference ones for fitting a face model on a facial region in the frame. Among the various methods of facial landmark detection, in recent years, random regression forest [24,25] and deep learning [20] algorithms have actively been used. Random regression forest-based methods are super-fast, yet highly dependent on initial guess, causing to erroneous detection. Deep learning-based methods provide robust detection, whereas the computational cost is high. One of main concerns in this study was accurate and robust facial landmark detection under various head poses and face shapes because registration errors are easily noticeable, so that the manipulation looks unnatural; in the proposed framework, we therefore take advantage of the deep learning-based method, proposed by Kowalski et al. [20]. Basically, this method builds a multistage deep neural network with several serially connected convolutional neural networks. At each stage, geometric transformation between input frame and the canonical shape of the face, the landmark heat map, and the feature map are estimated, and the locations of facial landmarks are refined by using them.

Based on the detected facial landmarks, a face model is fitted on a face region in the frame by updating and refining their corresponding landmarks on the model and head poses in an iterative manner. However, the face model fitting is quite challenging due to various facial shapes and expressions. To deal with them, the proposed framework adopts a 3DMM [21], which can represent various shapes and expressions, and further improves the face model fitting to allow real-time performance. Conventionally, the 3DMM is generated from a training set of 3D face scans and represented using principal component analysis (PCA) as a linear combination of basis vectors:

$$\mathbf{M} = \bar{\mathbf{M}} + \mathbf{M}_{sb} \cdot \boldsymbol{\alpha} + \mathbf{M}_{eb} \cdot \boldsymbol{\beta}, \tag{1}$$

where $\bar{\mathbf{M}}$ is the mean face model with neutral expression; $\mathbf{M}_{sb}$ and $\boldsymbol{\alpha}$ are the shape basis and coefficients; and $\mathbf{M}_{eb}$ and $\boldsymbol{\beta}$ are the expression basis and coefficients. Here, $\mathbf{M}$ is the 3D facial mesh, defined by a set of 3D vertices $\mathbf{X}$, $\mathbf{M} = [\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_n]^T \in \mathbb{R}^3$. Since the facial landmarks are annotated on the mean face model in advance, head pose can initially be estimated using the landmarks and their correspondences on the model. With the initial head pose, the shape coefficients can be computed by minimizing the following cost function, written by the reprojection error of the landmarks:

$$\tilde{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \frac{\left(\mathbf{P}\mathbf{X}_i\left(\boldsymbol{\alpha}\right) - \mathbf{x}_i\right)^2}{2\sigma^2} + \lambda \left\|\boldsymbol{\alpha}\right\|_2^2, \tag{2}$$

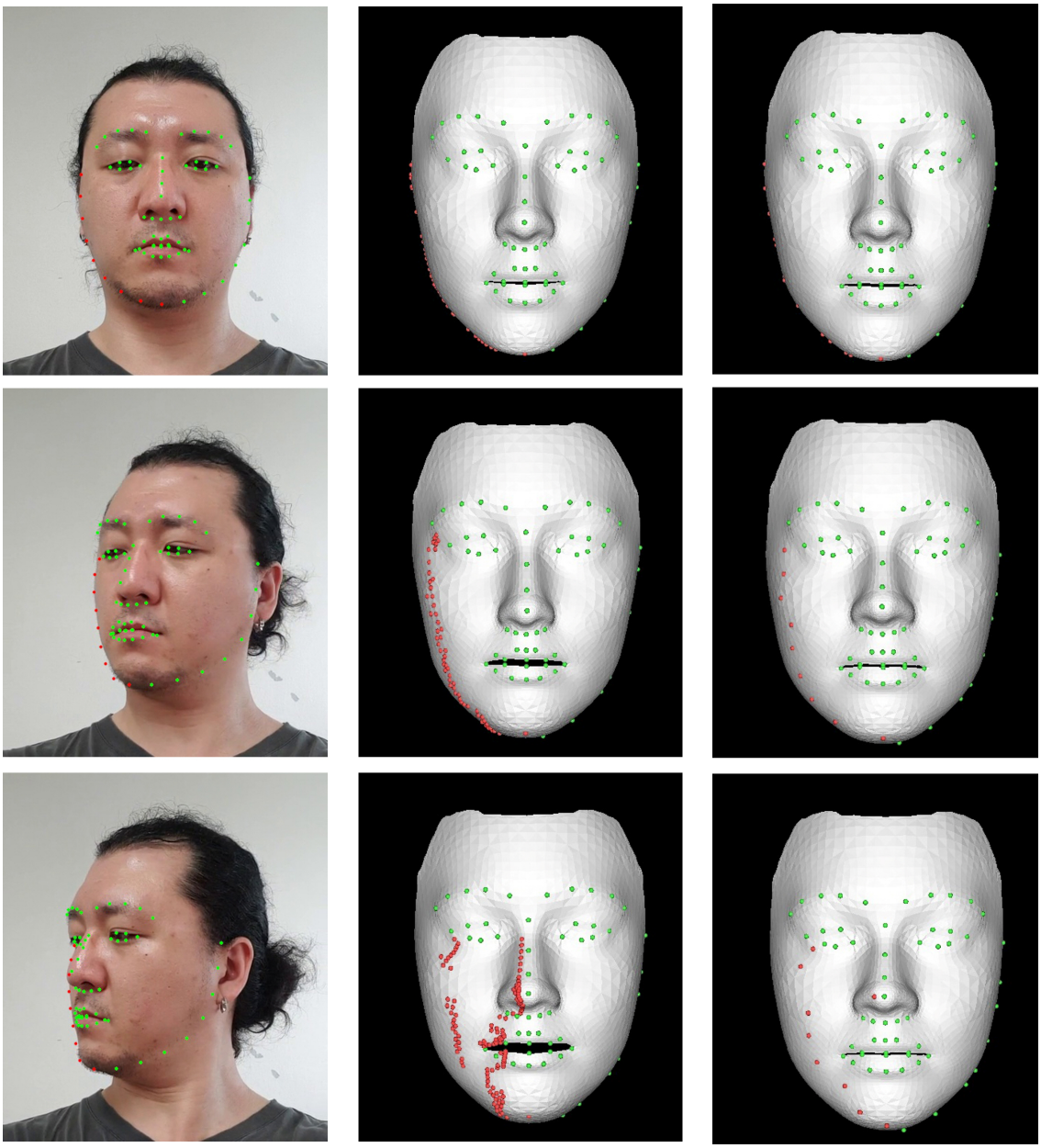
where N is the number of landmarks, σ is the standard deviation of landmarks, $\mathbf{P}$ is the camera projection matrix, $\mathbf{x}_i$ and $\mathbf{X}_i\left(\boldsymbol{\alpha}\right)$ are the ith facial landmark (2D position) detected from the video frame and the corresponding landmark (3D position) on the model, and λ is the regularization parameter. Here, the corresponding landmarks on the model $\mathbf{X}\left(\boldsymbol{\alpha}\right)$ are determined by the mean face model and the linear combination of the shape basis and coefficients:

$$\mathbf{X}\left(\boldsymbol{\alpha}\right) \subset \mathbf{M}_s, \ \mathbf{M}_s = \bar{\mathbf{M}} + \mathbf{M}_{sb} \cdot \boldsymbol{\alpha}. \tag{3}$$

The expression coefficients $\boldsymbol{\beta}$ are then computed using the estimated shape $\mathbf{M}_s$ in a similar way. Finally, head pose and coefficients are iteratively updated and refined.

As shown in Figure 3, however, the facial outline landmarks are not consistent for changes of the head pose; thus, the proposed framework searches directly for the corresponding landmarks on the model rather than using the predefined mapping between corresponding 2D and 3D landmarks. In [21], the vertices of the 3D face model, which belong to the front face from the current viewpoint and overlap others, are selected and projected on the video frame. Projected vertices within a certain distance of facial landmarks detected in the frame are defined as the corresponding landmarks on the model. In this approach, multiple landmarks are searched, especially when the viewpoint is oblique, which incurs a high computational cost in the aforementioned energy minimization. In Figure 3b, the correspondences of the 2D landmarks in Figure 3a are plotted on the 3D model; the number

of correspondences marked with red dots is much larger than the number of 2D ones. To tackle this problem, the proposed framework refines selected corresponding landmarks on the basis of several constraints. When multiple landmarks are found, the three closest vertices are pre-selected first. The intersection point between the backprojection ray of the landmark in the frame and the triangle mesh formed by the pre-selected three vertices is then computed and finally determined as the correspondence. As shown in Figure 3c, only one correspondence is taken for each landmark as a result of the correspondence searching method in the proposed framework.



(**a**) (**b**) (**c**)

Figure 3. Results of face model fitting: (**a**) face landmarks detected from the video frame; (**b**) corresponding landmarks on the model searched by Huber et al. [21]; and (**c**) corresponding landmarks on the model searched by the proposed framework. Green dots indicate facial landmarks, and red dots highlight facial outline landmarks.

2.2. Manipulation

After the face model has been fitted on the video frame, it can be deformed by the user's input, based on the ARAP method [22]. ARAP is a mesh deformation method that allows natural deformation while maintaining the rigidity of the local surface through a two-step optimization with enforced rotational transformation. Given a mesh model $\mathbf{M}$ with n vertices, deformation energy E can be formulated in terms of its vertices $\mathbf{X}_i,\ i \in \{1, ..., n\}$ and their neighbors $\mathbf{X}_j \in N(\mathbf{X}_i)$ as

$$E = \sum_{\mathbf{X}_i \in \mathbf{M}} \sum_{\mathbf{X}_j \in N(\mathbf{X}_i)} \omega_{ij} \left\| \left(\mathbf{X}'_i - \mathbf{X}'_j \right) - \mathbf{R}_i \left(\mathbf{X}_i - \mathbf{X}_j \right) \right\|^2, \tag{4}$$

where ω_{ij} is the weight, $\mathbf{X}'_i$ and $\mathbf{X}'_j$ are the positions of $\mathbf{X}_i$ and $\mathbf{X}_j$ after the deformation, and $\mathbf{R}_i$ is the rotation matrix of the local surface defined by $\mathbf{X}_i$ and $N(\mathbf{X}_i)$. In the two-step optimization, $\mathbf{X}'_i$ and $\mathbf{X}'_j$ are unknown variables to be estimated, playing different roles as three types of anchor constrained by a user:

- static anchors (vertices to be fixed);
- handle anchors (vertices to be moved by a user); and
- ROI anchors (vertices of the region of interest, to which the deformation is propagated).

In the first step of optimization, the rotation matrices for the local surfaces are computed with the positions of ROI anchors being fixed. Then, the rotation matrices are fixed, and the positions of the moving vertices are calculated by minimizing the cost function. Figure 4 shows our results for mesh deformation by a user's input. On the original mesh model (Figure 4a), handle anchors and ROI anchors were set by the user (Figure 4b), and the model could instantly be deformed when the positions of the handle anchors were changed by the user (Figure 4c,d).

Basically, an anchor is taken by picking a vertex. When a user specifies 2D screen coordinates by clicking a mouse, the 2D coordinates are backprojected and converted to a ray. Then, a facet is selected by computing the intersection between the ray and the model, and the closest vertex which belongs to the facet will be the anchor. In other words, when a spot on a face model is chosen by a user, its closest vertex of the model is defined as an anchor. Although every vertex on the model can be the handle or the ROI anchors, it can lead to discontinuities in the borders with areas that the face model doesn't cover.

On the other hand, the face model fitted in the registration module is not suitable for selecting and moving the handle anchors because it is changed for every frame according to face viewpoint and expression. For that reason, the proposed framework performs the deformation on the mean face model in the neutral state rather than directly using the fitted face model. In this regard, the face model in Equation (1) is rewritten by an additional term for the final rendering:

$$\mathbf{M}' = \left(\bar{\mathbf{M}} + \mathbf{M}_{sb} \cdot \tilde{\boldsymbol{\alpha}} + \mathbf{M}_{eb} \cdot \tilde{\boldsymbol{\beta}} \right) + \mathbf{W} \left(\mathbf{M}_m - \bar{\mathbf{M}} \right), \tag{5}$$

where $\mathbf{M}_m$ is the deformed model in the manipulation module and $\mathbf{W}$ is the weight matrix that controls the degree of deformation of each vertex. Here, the weights for every vertices in the $\mathbf{W}$ are set to 1 in all cases, and the user deformation is applied as is—that is, the model is determined by merging the fitted model with the difference between after and before mesh deformation.

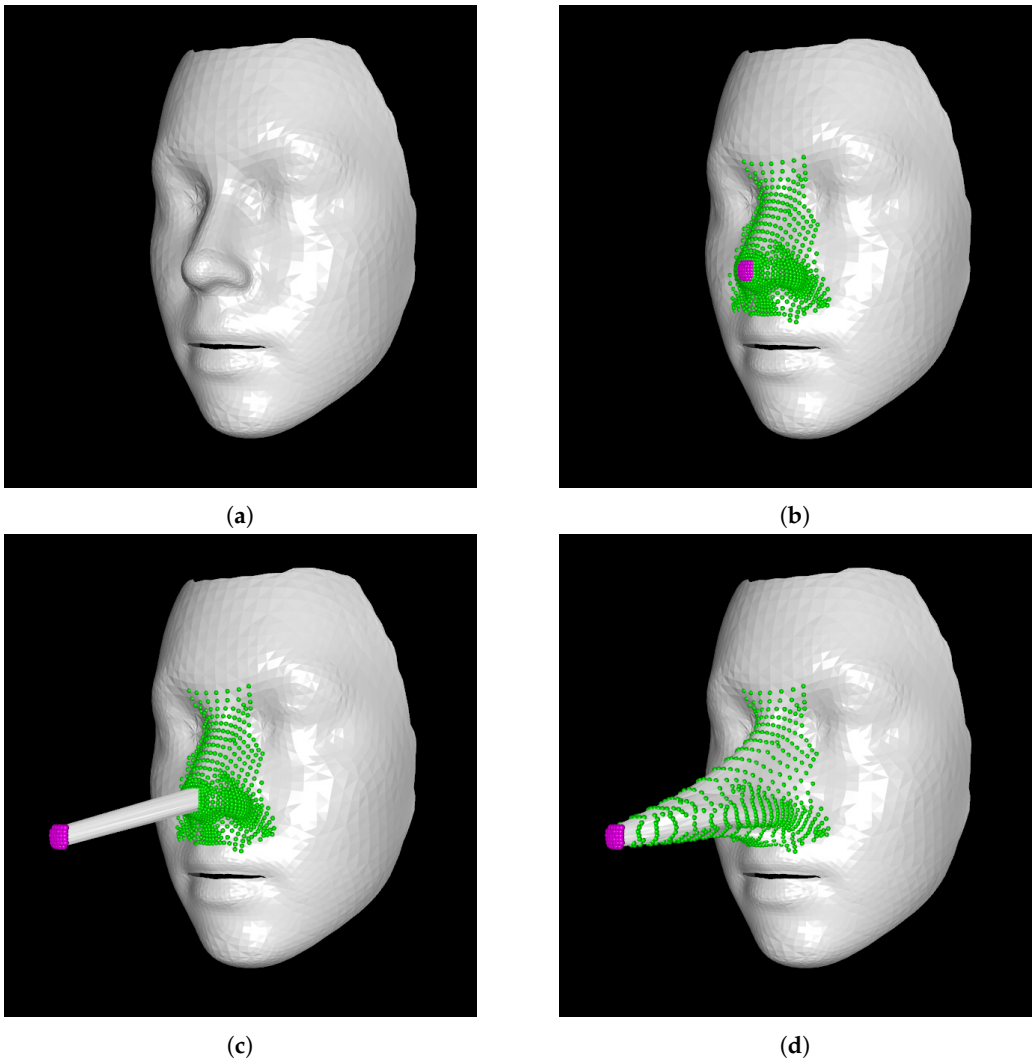


Figure 4. Results of mesh deformation by a user's input: (**a**) original mesh model; (**b**) setting handle anchors (purple dots) and ROI anchors (green dots) as a user constraint; (**c**) moving handle anchors; and (**d**) deformed mesh.

2.3. Rendering

In the rendering module, the final image is generated simply by composing the video frame and the deformed facial model. In some cases, however, the hidden regions appear in the frame behind the face region; for example, the face region may shrink partially after deformation, and its hidden regions may cause a visual intrusion for the user. To remove such artifacts, the proposed framework applies an image inpainting method. The mask to be used in the inpainting is obtained by projecting the face model on the image plane. Diffusion-based inpainting method fills the masked region from boundary to inside by referring to the outward neighbors of the destination pixels [26]. Note that this method makes it difficult to fill the hidden regions perfectly in real time when they are large or highly textured; in the proposed framework, however, it is assumed that the natural deformation is applied to the face but not the head and neck, which means that the regions to be actually filled are not unduly large. Finally, after rendering the inpainted frame, the texture map of the face model is updated with the current frame using the projection transformation between the model and frame, and the deformed model is then rendered on the inpainted frame.

3. Experimental Results

3.1. Implementation

The application programming interface of the proposed framework was designed using Python. Facial landmark detection was implemented in Python using Theano, a deep learning framework, and other functions such as face model fitting and mesh deformation were implemented in C++ and invoked via Python bindings. Partial parallel processing with OpenMP was applied where extensive computation was required, as for example in visibility checking of all vertices of the face model to search for corresponding 3D landmarks on the model of 2D facial landmarks along the boundary of the face.

In the proposed framework, moreover, each module was assigned to separate threads for more speed-up, allowing the entire framework to operate in real time. However, it was difficult for each module to operate completely in parallel because of data sharing requirements. For instance, the manipulation module needed the results of fitting from the registration module, and the rendering module needed both the camera pose from the registration module and the deformed model from the manipulation module. To handle such dependencies more efficiently, the manipulation and rendering modules instantly used data from the previous frame rather than waiting those from the current frame. Figure 5 illustrates our inter-module parallelization scheme. While the registration and rendering modules perform their processes synchronously with the frame (orange and gray rectangles), in the manipulation module, processes run asynchronously based on a user's input regardless of frame (green rectangles). Whenever a user picks a handle anchor with the mouse and moves the mouse pointer, the mouse events are pushed to a queue, and the processes in the manipulation module run repeatedly while emptying the queue.

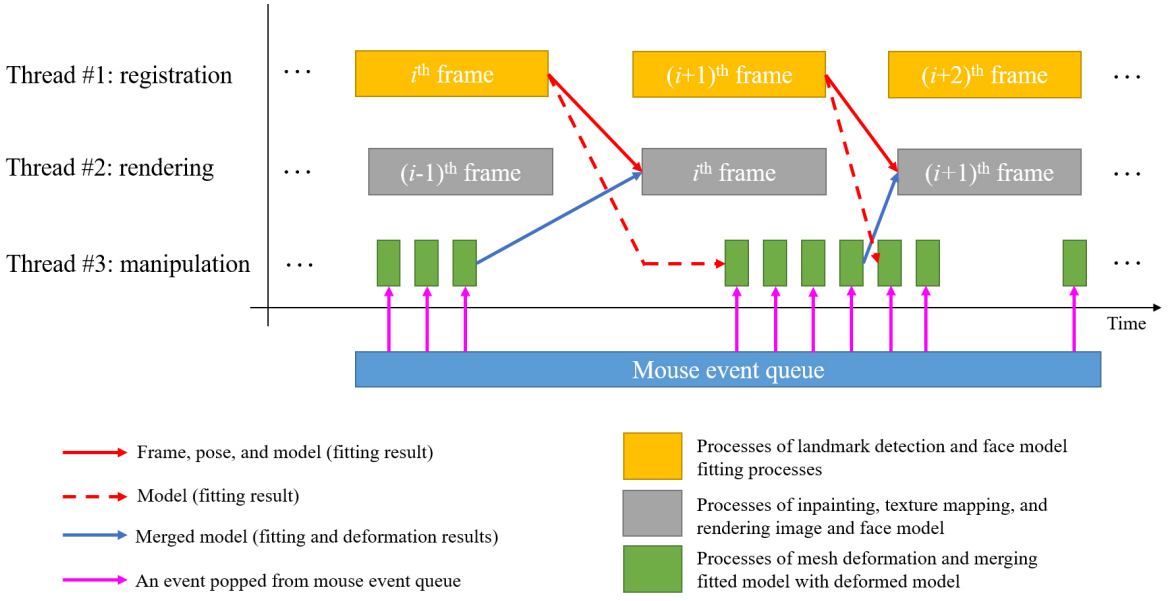


Figure 5. Inter-module parallelization scheme.

3.2. Manipulation Results

We performed experiments to demonstrate video manipulation (Figure 6) and augmented reality (Figure 7) using the proposed framework; the Supplementary Material shows these demonstrations. Figure 6 shows the results of manipulation of the original face (Figure 6a) in video frames (see the Supplementary Materials) using the proposed framework. When the face model was deformed by the user's input (as shown in the first column of Figure 6), the shape of the face in the frame was naturally deformed without background distortion (second to sixth columns). However, some artifacts

occurred at the boundary of the face (e.g., the jaw in the fourth and fifth columns of Figure 6c,e) because the background pixels were incorporated into the texture of the deformed surface as a result of registration errors. Possibly, inpainting errors could occur when the occluded regions were large due to heavy deformation. Figure 7 shows an example of augmented reality application using the proposed framework where a virtual hand picks and stretches nose of a real person.

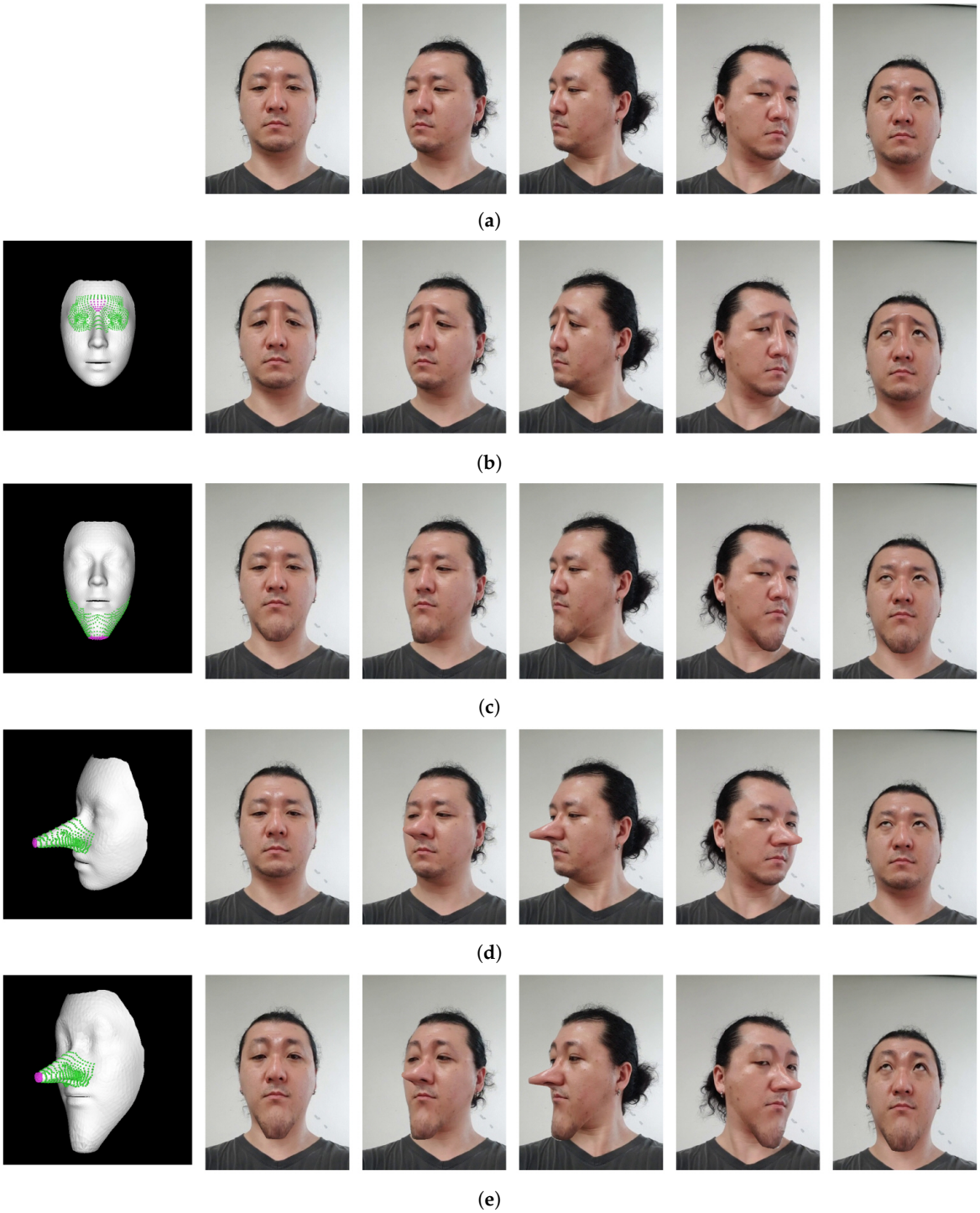


Figure 6. Results of face manipulation for video frames: (**a**) original frames; (**b**–**d**) results of manipulating specific parts of face; and (**e**) results of applying all manipulations in (**b**–**d**).

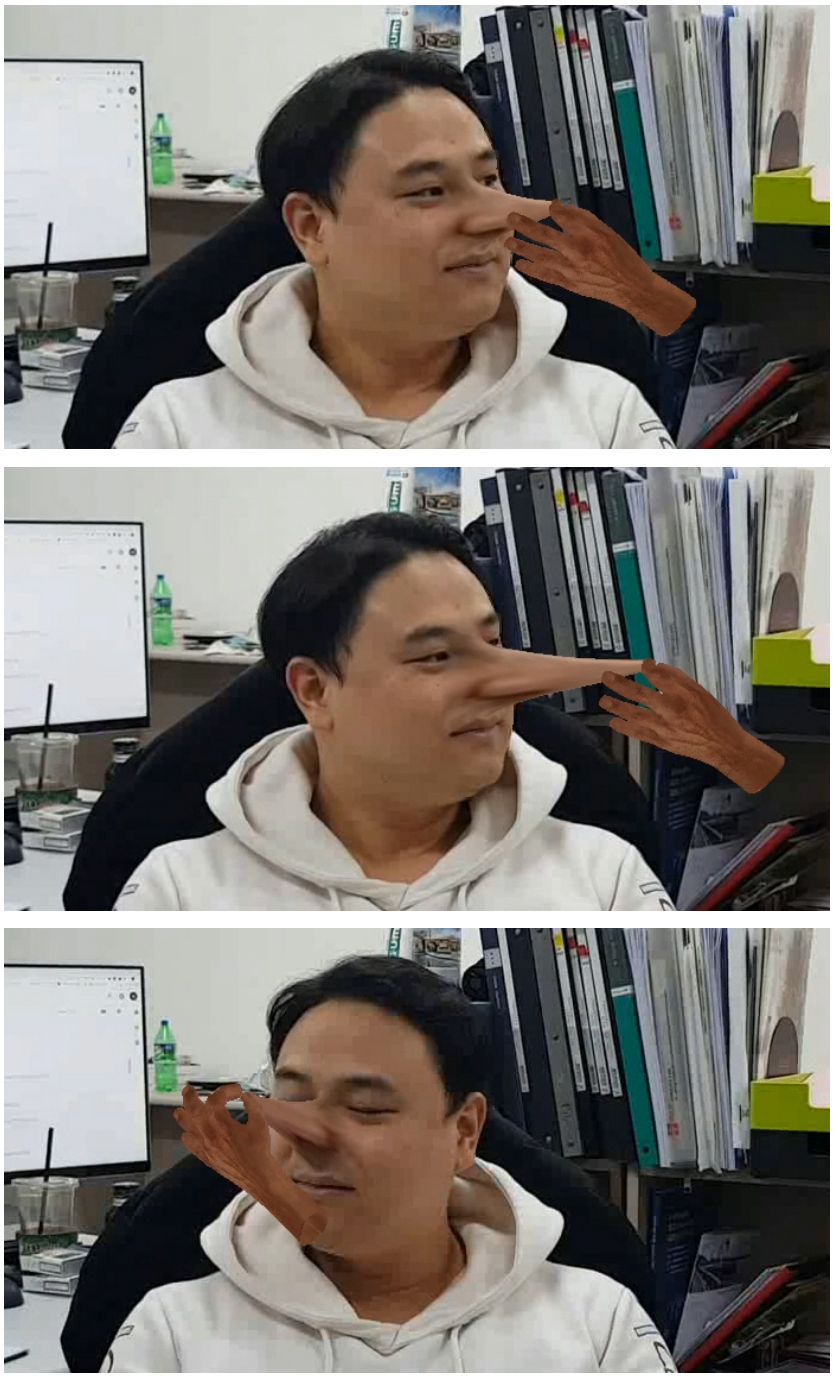


Figure 7. An example of augmented reality application using the proposed framework.

3.3. Performance Evaluation and Comparison

To verify real-time performance, we evaluated the processing time of the proposed framework on a desktop PC (Intel i7-3770 CPU and NVidia GeForce GTX TITAN GPU) using a face model with 3448 vertices and 6736 triangles and a video with resolution of 856 × 480. At first, for the frontal and oblique face, processing times in face model fitting were computed with and without two acceleration schemes: parallel processing with OpenMP and correspondence filtering (as described in the last paragraph of Section 2.1). As shown in Table 1, processing time was sped up in every case, regardless of head pose. For the oblique face in particular, processing time was drastically dropped by about 37 ms with OpenMP and 29 ms with correspondence filtering. Note that, since neither acceleration affected whole face model fitting process, processing times were not proportional,

regardless of acceleration applied. Even though face model fitting functioned in real time with acceleration, the overall performance of the framework might suffer from other heavy tasks such as landmark detection, mesh deformation, and inpainting. As shown in Table 2, total processing time was about 113 ms per frame (9 fps) on the assumption that entire processes were sequentially performed, which might not be sufficient for real-time applications. Moreover, processing time may be further increased because mesh deformation can run several times within one frame when the user is moving handle anchors. In the proposed framework, however, inter-module parallelization scheme enabled carrying out the entire processes at about 15 fps, derived from the least upper bound (58.49 ms) of subtotal processing times.

Table 1. Processing time (ms) for face model fitting without and with acceleration.

	Without Correspondence Filtering		**With Correspondence Filtering**	
	Without OpenMP	**With OpenMP**	**Without OpenMP**	**With OpenMP**
Frontal face	35.70	25.77	30.15	20.63
Oblique face	97.15	60.31	68.03	30.58

Table 2. Total processing time (ms).

Module	**Process**	**Time**	**Subtotal Time**
Registration	Facial landmark detection	36.28	58.49
	Face model fitting	22.21	
Manipulation	Mesh deformation	7.14	12.48
	Merging fitted and deformed models	5.34	
Rendering	Inpainting	31.16	41.71
	Update textures	3.37	
	Update texture coordinates	6.49	
	Rendering image	0.36	
	Rendering face model	0.33	
Total			112.68

For qualitative evaluation, we compared our manipulation results with two existing applications: Plastic Surgery Simulator [4] (Figure 8) and Snow [5] (Figure 9). In the former case, it was difficult to manipulate the face precisely and consistently because the deformation was performed manually for each frame by stretching or crushing given point of the 2D frame and its surrounding area; thus, as shown in Figure 8c, artifacts were incurred in several deformed parts (e.g., stretched nose), and background regions were distorted around the neck or t-shirt. In Snow, manipulation was often accompanied by background distortion. Natural manipulations were possible only for the frontal face and became worse with head pose changes (e.g., turning the face sideways or leaning forward), as shown in Figure 9a. Moreover, only a few predefined deformations could be selected and applied. Different from them, our results show that natural 3D face manipulation could be freely done to desired shapes at any head pose without background distortion.

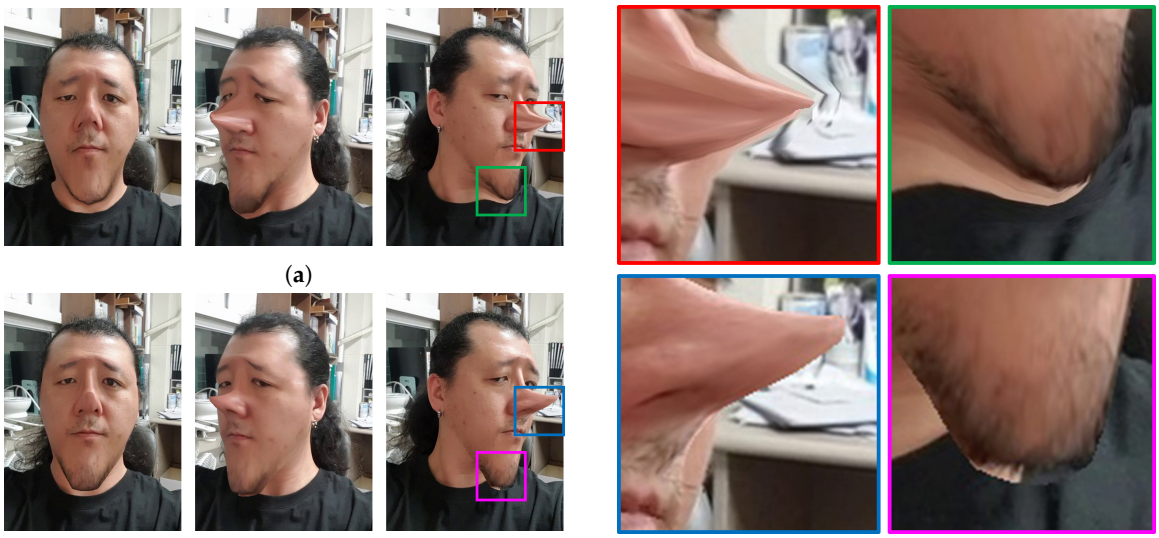


Figure 8. Comparison with existing application: (**a**) results from Plastic Surgery Simulator; (**b**) results from the proposed framework; and (**c**) magnifications of the color boxed in (**a**,**b**).

Figure 9. Comparison with existing application: (**a**) results from Snow; and (**b**) results from the proposed framework.

4. Conclusions

This paper describes a new framework for 3D freeform manipulation of a face in video frames. The proposed framework estimated the states of the 3D face model, which were interactively deformed by the user's input and applied to current state of the face model, allowing the user to manipulate the

face freely and naturally. Additionally, the parallelization and acceleration schemes enabled real-time deformation. The feasibility of the proposed framework was verified by performance evaluation and comparison with existing applications. Moreover, the example of the nose being stretched by the virtual hand shows that the proposed framework could be applied to various augmented reality applications as well as video editing applications.

Although the proposed framework allows 3D manipulation of a face without background distortion, there are some limitations to be carefully considered. For instance, some artifacts may occur at the edges of the deformed face due to registration errors or inpainting errors because the proposed framework uses the current frame as the texture of the face model. Furthermore, this can result in the texture of the invisible parts from the current viewpoint not being represented correctly when they are revealed by manipulation. As a future work, we plan to investigate high-quality face editing by tackling degradation cases such as untextured parts and artifacts due to registration errors and inpainting errors.

Supplementary Materials: The following are available online at http://www.mdpi.com/2076-3417/9/21/4707/s1, Video S1: A Framework for Real-time 3D Freeform Manipulation of Facial Video.

Author Contributions: Conceptualization, B.-K.S. and J.-I.P.; Funding acquisition, J.-I.P.; Investigation, J.P.; Methodology, J.P., B.-K.S. and J.-I.P.; Project administration, J.-I.P.; Software, J.P.; Supervision, J.-I.P.; Writing—original draft, J.P.; and Writing—review and editing, J.P., B.-K.S. and J.-I.P.

Funding: This work was supported by Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.2017-0-01849, Development of Core Technology for Real-Time Image Composition in Unstructured In-outdoor Environment)

Conflicts of Interest: The authors declare no conflict of interest.

References

1. FaceApp - Free Neural Face Transformation Filters. Available online: https://faceapp.com (accessed on 12 September 2019).
2. Face Warp—Plastic Surgery. Available online: https://play.google.com/store/apps/details?id=com.hamsoft.face.follow (accessed on 12 September 2019).
3. Wombatica Software—Create And Share. Available online: https://wombatica.com (accessed on 12 September 2019).
4. Virtual Plastic Surgery Simulator. Available online: https://www.plastic-surgery-simulator.com (accessed on 12 September 2019).
5. Snow. Available online: https://snow.me (accessed on 12 September 2019).
6. Chung, H.V.; Lee, I.K. Image-based deformation of objects in real scenes. In *International Symposium on Visual Computing*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 159–166.
7. Park, J.; Seo, B.; Park, J. [Poster] Interactive deformation of real objects. In Proceedings of the 2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), Munich, Germany, 10–12 September 2014; pp. 295–296. doi:10.1109/ISMAR.2014.6948457. [CrossRef]
8. Zheng, Y.; Chen, X.; Cheng, M.M.; Zhou, K.; Hu, S.M.; Mitra, N.J. Interactive Images: Cuboid Proxies for Smart Image Manipulation. *ACM Trans. Graph.* **2012**, *31*, 99:1–99:11. doi:10.1145/2185520.2185595. [CrossRef]
9. Chen, T.; Zhu, Z.; Shamir, A.; Hu, S.M.; Cohen-Or, D. 3Sweepp: Extracting Editable Objects from a Single Photo. *ACM Trans. Graph.* **2013**, *32*, 195:1–195:10. doi:10.1145/2508363.2508378. [CrossRef]
10. Kholgade, N.; Simon, T.; Efros, A.; Sheikh, Y. 3D Object Manipulation in a Single Photograph Using Stock 3D Models. *ACM Trans. Graph.* **2014**, *33*, 127:1–127:12. doi:10.1145/2601097.2601209. [CrossRef]
11. Haouchine, N.; Petit, A.; Roy, F.; Cotin, S. [POSTER] Deformed Reality: Proof of Concept and Preliminary Results. In Proceedings of the 2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct), Nantes, France, 9–13 October 2017; pp. 166–167. doi:10.1109/ISMAR-Adjunct.2017.56. [CrossRef]
12. Thies, J.; Zollhöfer, M.; Stamminger, M.; Theobalt, C.; Nießner, M. Face2Face: Real-Time Face Capture and Reenactment of RGB Videos. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern

Recognition (CVPR), Las Vegas, NV, USA, 26 June–21 July 2016; pp. 2387–2395. doi:10.1109/CVPR.2016.262. [CrossRef]
13. Dale, K.; Sunkavalli, K.; Johnson, M.K.; Vlasic, D.; Matusik, W.; Pfister, H. Video Face Replacement. *ACM Trans. Graph.* **2011**, *30*, 130:1–130:10. doi:10.1145/2070781.2024164. [CrossRef]
14. Kitanovski, V.; Izquierdo, E. Augmented reality mirror for virtual facial alterations. In Proceedings of the 2011 18th IEEE International Conference on Image Processing, Brussels, Belgium, 11–14 September 2011; pp. 1093–1096. doi:10.1109/ICIP.2011.6115616. [CrossRef]
15. Cao, C.; Weng, Y.; Zhou, S.; Tong, Y.; Zhou, K. FaceWarehouse: A 3D Facial Expression Database for Visual Computing. *IEEE Trans. Vis. Comput. Graph.* **2014**, *20*, 413–425. doi:10.1109/TVCG.2013.249. [CrossRef] [PubMed]
16. Saragih, J.M.; Lucey, S.; Cohn, J.F. Real-time avatar animation from a single image. *Face Gesture* **2011**, *2011*, 117–124. doi:10.1109/FG.2011.5771383. [CrossRef]
17. Weng, Y.; Cao, C.; Hou, Q.; Zhou, K. Real-time facial animation on mobile devices. *Graph. Model.* **2014**, *76*, 172–179. doi:10.1016/j.gmod.2013.10.002. [CrossRef]
18. Cao, C.; Wu, H.; Weng, Y.; Shao, T.; Zhou, K. Real-time Facial Animation with Image-based Dynamic Avatars. *ACM Trans. Graph.* **2016**, *35*, 126:1–126:12. doi:10.1145/2897824.2925873. [CrossRef]
19. Nagano, K.; Seo, J.; Xing, J.; Wei, L.; Li, Z.; Saito, S.; Agarwal, A.; Fursund, J.; Li, H. paGAN: Real-time Avatars Using Dynamic Textures. *ACM Trans. Graph.* **2018**, *37*, 258:1–258:12. doi:10.1145/3272127.3275075. [CrossRef]
20. Kowalski, M.; Naruniec, J.; Trzcinski, T. Deep Alignment Network: A Convolutional Neural Network for Robust Face Alignment. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, Honolulu, HI, USA, 21–26 July 2017.
21. Huber, P.; Hu, G.; Tena, R.; Mortazavian, P.; Koppen, P.; Christmas, W.J.; Ratsch, M.; Kittler, J. A multiresolution 3D morphable face model and fitting framework. In Proceedings of the 11th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, Rome, Italy, 27–29 February 2016.
22. Sorkine, O.; Alexa, M. As-rigid-as-possible Surface Modeling. In Proceedings of the Fifth Eurographics Symposium on Geometry Processing (SGP '07), Barcelona, Spain, 4–6 July 2007; Eurographics Association: Aire-la-Ville, Switzerland, Switzerland, 2007; pp. 109–116.
23. Park, J.; Park, J. A Framework for Virtual 3D Manipulation of Face in Video. In Proceedings of the 2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR), Reutlingen, Germany, 18–22 March 2018; pp. 649–650. doi:10.1109/VR.2018.8446445. [CrossRef]
24. Ren, S.; Cao, X.; Wei, Y.; Sun, J. Face Alignment at 3000 FPS via Regressing Local Binary Features. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Columbus, OH, USA, 23–28 June 2014.
25. Kazemi, V.; Sullivan, J. One Millisecond Face Alignment with an Ensemble of Regression Trees. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Portland, OR, USA, 23–28 June 2014.
26. Telea, A. An Image Inpainting Technique Based on the Fast Marching Method. *J. Graph. Tools* **2004**, *9*, 23–34. doi:10.1080/10867651.2004.10487596. [CrossRef]