Digital Facial Augmentation for Interactive Entertainment

Naoto Hieda and Jeremy R. Cooperstock
Centre for Intelligent Machines
McGill University
Montreal, Quebec, Canada
Email: {nhieda, jer}@cim.mcgill.ca

***Abstract*—Digital projection technology allows for effective and entertaining spatial augmented reality applications. Leveraging the capabilities of reasonably accurate object tracking using commodity cameras and/or depth sensors to determine the 3D position and pose of objects in real time, it is possible to project dynamic graphical content on arbitrary surfaces, such as a person's face. Coupling these capabilities with a simple drawing application, participants can have the experience of "painting" on someone's face, or even on their own, by observing the projection in a mirror. Similarly, integrating 2D rigid-body, fluid and gravity simulation, one may interact with virtual objects projected on their own face or body.**

Keywords*—*projection mapping; augmented reality.

I. Introduction

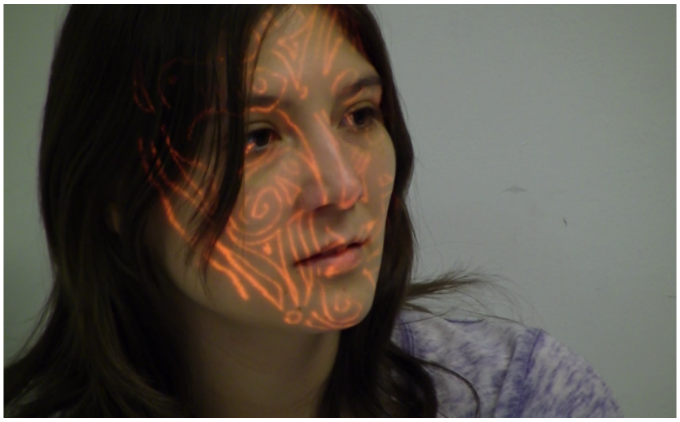


Fig. 1. A tattoo-like pattern projected on a face.

Drawing on a face, in the form of "make-up", represents an important element of entertainment. It offers playful, ludic benefits, e.g., face painting to transform a child into an animal, and even has cultural associations, for example with aboriginal art used in ceremonies. Although such drawing has traditionally relied on the physical application of paint, it is possible to map digital textures to the face using a combination of video projection and face tracking technologies. Significantly, this approach need not be limited to the application of one-shot face painting, i.e., projection of a static texture, but can be used to project time-varying, dynamic contents on the face, often producing compelling aesthetic results [1], [2]. These examples, however, are designed for art performance. We are unaware of other systems that consider the possibilities of interaction with the projected face. This paper describes our exploration of the potential of face projection technology for interactive applications, illustrated by several examples we developed for entertainment purposes.

II. Literature Review

Spatial augmented reality often involves real-time object tracking and projection of "spatially aware" contents onto a tabletop or wall surface [3]. Extending this approach as a public installation, MobiSpray employs interactive components for artistic effect, transforming a smartphone into a virtual spray can to draw on building facades or monuments [4]. Similar effects have been achieved in accurately mapping video contents onto a human body or face, using depth cameras or motion capture systems for face tracking [1], [2]. Although the motivation for projection onto the face is obvious given that it is the most prominent part of the body, there are significant technical challenges posed by latency, the uniqueness of each individual, and the deformability of the human face. Using the recent face-tracking algorithm of Cai et al. [5], which accounts for such deformation, we developed an interactive facial projection mapping system that allows for "painting" and interacting with physical simulations, all projected on peoples' faces.

In the remainder of this section, we review object tracking methods, video projection systems, and artistic and entertainment applications of these technologies.

A. Object Tracking

Object tracking for augmented reality systems typically requires locating and tracking the objects of interest, and overlaying contents on their physical surfaces by projection, preferably in real-time. Common methods include depth cameras and motion-capture systems.

Depth Camera: Using a projector-camera pair, one-shot structured light [6] can obtain the depth image in real-time and is thus able to update the 3D model. Commodity depth cameras are based on the one-shot technique, with a projector replaced by a laser emitter for robust geometry acquisition. Nonetheless, the output is limited to a set of 3D points, which does not directly provide the information of the objects in the scene (e.g., the pose or label of an object). Such information is often desired for systems that warp video contents using the position and orientation of the target and accurately project on its surface. When a depth camera is used, the target pose is

INTETAIN 2015, June 10-12, Torino, Italy

DOI 10.4108/icst.intetain.2015.259444

only acquired by fitting a pre-modeled 3D mesh to the scanned depth image using methods such as iterative closest point and particle filtering. However, this method is computationally expensive and ill-posed, requiring use of GPU acceleration to be tractable [7]. To address the challenges for human tracking, Cai et al. [5] developed a face-tracking algorithm that works with a commodity depth camera, the Microsoft Kinect, and takes account of individuality and deformation.

Marker Tracking: Color cameras are suitable for object tracking in real-time applications, exploiting fiducials [3] or an optical model of the object surface [8] to estimate the relative pose between the projector and target. Although many systems project contents on planar or quadratic surfaces using homography or quadratic parameterization to warp the contents, these typically assume a fairly simple surface structure. To achieve projection on more complex structures, motion capture systems are often used. These employs IR or near-IR LEDs to illuminate retro-reflective markers attached to the target. Using multiple IR-sensitive cameras, the marker positions can then be triangulated to obtain the 3D position information. Since the target pose can be inferred from the marker positions, projection on any target is possible when the target is a rigid body and its surface geometry is known, for example, a face [2]. Similarly, structured lighting can be integrated to a marker-tracking system by affixing photosensors to the target and locating the photosensors by decoding the received codewords [9], [10].

B. Video Projection Systems

Video projection is popular for use in virtual reality immersive display environments, because it is more easily scalable than fixed-size liquid-crystal displays (LCDs). Moreover, it is possible to combine multiple projectors with blending techniques, and illuminate multiple surfaces simultaneously [11]. Gamma correction of overlapping projected regions and surface color compensation can be achieved simultaneously, making use of raw camera images of the projected output, captured at different shutter speeds [12].

In augmented reality research, one of the earliest video projection applications unconstrained by a tabletop is known as an object-adaptive display, introduced by Raskar et al. [3]. With a handheld device consisting of a calibrated projector-camera pair, the camera captures a fiducial, or *piecode*, similar to a piechart segmented by different colors, in a physical environment so that the label and pose of the fiducial can be determined. Since the projector is calibrated to the camera, the fiducial in the projector image plane is estimated, and video texture can be projected on the physical surface. Lee et al. [9] used a fixed projector, and photosensors affixed to an object to reliably locate the object pose for video projection, described as marker tracking in Section II-A.

Unlike most CAVE-like systems that treat the environment as a screen on which the users focus, the IllumiRoom [13] demonstrated projection around the periphery of a television screen, i.e., the walls and furniture. In this way, the projection is not intended to attract the users' focus, but to achieve an optical illusion or enhance the user experience. To acquire the 3D geometry surrounding the television, a Kinect is calibrated to the projector by gray-coded structured lighting. The point cloud captured by the Kinect is then warped to the projector coordinates for video mapping.

C. Artistic and Entertainment Applications

As opposed to interactive virtual and augmented reality systems, artists have also made extensive use of projection-mapping techniques for art installations, often non-interactive. A common example is that of projector mapping onto prominent building facades such as the Sagrada Familia, Tokyo Station, and the Sydney Opera House, to change the appearance of their texture or shape.

MobiSpray [4], noted above, couples projection mapping with user interaction, allowing users to optically paint building facades by projection controlled by smartphones. Facial recognition [14] by a video camera combined with a Kinect was used to track the performer's face in order to support facial projection mapping, with unrestricted motion, in Bell's music video "Chase No Face" [1]. This tracking approach relies only on a 2D RGB stream, i.e., color images, and uses depth data for mesh recovery but not for tracking. Since the face detection is performed on a 2D image, the number of vertices it obtains of the face mesh is limited (in this case, to 66), whereas the method by Cai et al. using a Kinect point cloud and the iterative closest point algorithm outputs a finer mesh (121 vertices) [5]. The more detailed mesh provided by the latter approach is better suited to supporting complex textures, which can be distorted when projected onto the face.

Facial projection mapping can be categorized into two approaches: texture-mapping the entire face, or facial augmentation of selected features. The former is intended to change the appearance of the entire face, for example, overlaying an animal face instead. This requires measurement of the spectral reflectance of the face texture, and compensating for the original texture in the projected image [15]. This is non-trivial, since the projection target may move constantly. To simplify the problem, white foundation makeup may be applied to the face [16], [2], but this is infeasible for interactive systems in exhibitions. Alternatively, face masks can be worn, but these have the undesirable effect of hiding the face, which eliminates its individuality.

Augmenting the face, in contrast, involves the addition of projected local features over the original face texture. For instance, a cheek color can be exaggerated, or a virtual tattoo can be mapped. This approach is more feasible since the projected texture can be perceived correctly without color compensation.

III. Implementing a Facial Augmentation System

In this section, we describe our initial efforts to develop a facial augmentation approach by adding local features. The system uses a depth camera to track the motion of an individual, wearing goggles. Other participants use a digital tablet to draw pictures that are streamed to the video renderer and projected on the physical face by means of a calibrated short-throw projector. Thus, the participants can see their drawings appear on someone's face in real time through projection mapping, without requiring any extra devices. The individual whose face is "drawn upon" can observe the projection through a mirror.

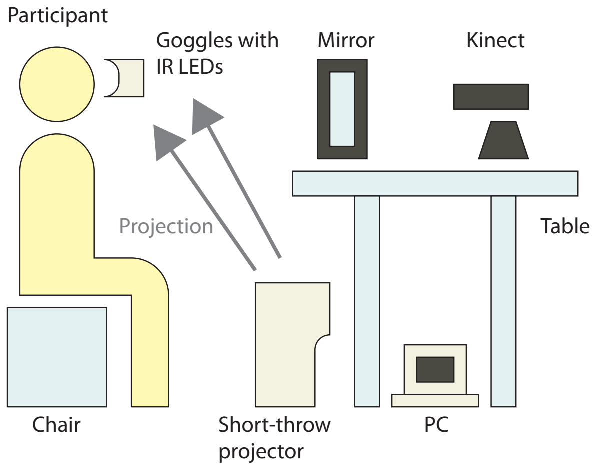


Fig. 2. Side view of the first prototype.

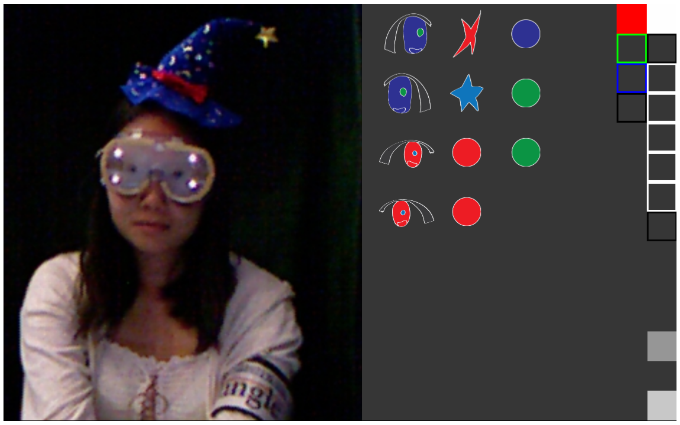


Fig. 3. Tablet interface. On the left half of the image, a participant with the goggles is shown. On the right half, stickers are displayed along with pen color selection buttons.

Before projection mapping, the hardware must be calibrated (Section III-A), and a 3D representation of the face, with goggles, must be acquired, with the depth camera operated in a color image acquisition mode. The data is segmented by eliminating points that are more than 30 cm from the markers, since these are guaranteed to be outside of the confines of the face, and a 3D mesh is formed by Delaunay triangulation [17]. The depth camera is then operated in IR image acquisition mode, thereby tracking the IR LED markers (Section III-B). To compensate for the latency of the system, a prediction filter estimates the position of the user's face (Section III-C). Once the marker coordinates are estimated, the pose of the face is computed by least-squares fitting [18]. The 3D mesh with an updated pose is rendered in the projector framebuffer, with texture as specified by the drawing application (Section III-D).

A. *Calibrating the Hardware*

Our first prototype consists of a 1024 × 768 pixel resolution short-throw projector, goggles with markers, a mirror, a Microsoft Kinect for Xbox 360 depth camera, a tablet device and a PC connected to a WiFi network, as illustrated in Figure 2. Four wide-viewing-angle IR LED markers are affixed to the goggles, allowing the participant's face to be tracked by the depth camera. Although the Kinect SDK provides a face-tracking library, the use of goggles is also necessary to protect the individual's eyes from the bright projection light. Otherwise, masking of the texture in the vicinity of the eyes would be necessary.

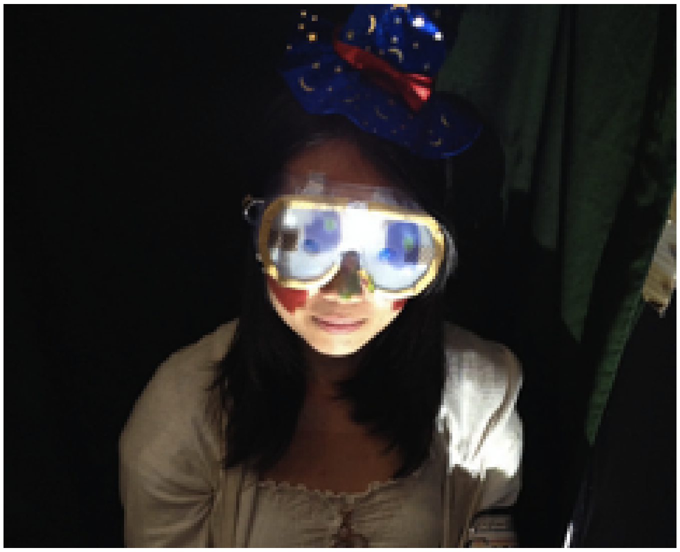


Fig. 4. A photo taken during the demonstration at International Collegiate Virtual Reality Contest 2014.

As a preliminary step, the projector and depth camera are calibrated to obtain the intrinsic and extrinsic camera parameters necessary to align the projection. Intrinsic calibration of the projector is performed by ProCamCalib [19], using a 1032 × 776 pixel resolution PointGrey Flea2 camera as a reference to find the focal length and lens distortion coefficients of the projector. Next, the depth camera is calibrated with the projector by gray-code structured lighting [6] to find pixel correspondences. The relative pose between the projector and camera is estimated by Levenberg-Marquardt fitting, using the OpenCV camera calibration [20] API. Let $\mathbf{M_i}$ denote a 3D point output by the depth camera. The reprojected 2D coordinate $\mathbf{m_i}$ of the point $\mathbf{M_i}$ on the projector image plane is

$$\mathbf{m_i} = \mathbf{K}\ \mathbf{E}\ \mathbf{M_i} \tag{1}$$

where the intrinsic matrix $\mathbf{K}$ is known, and the extrinsic matrix has six unknowns for rotation and translation. The reprojection error is

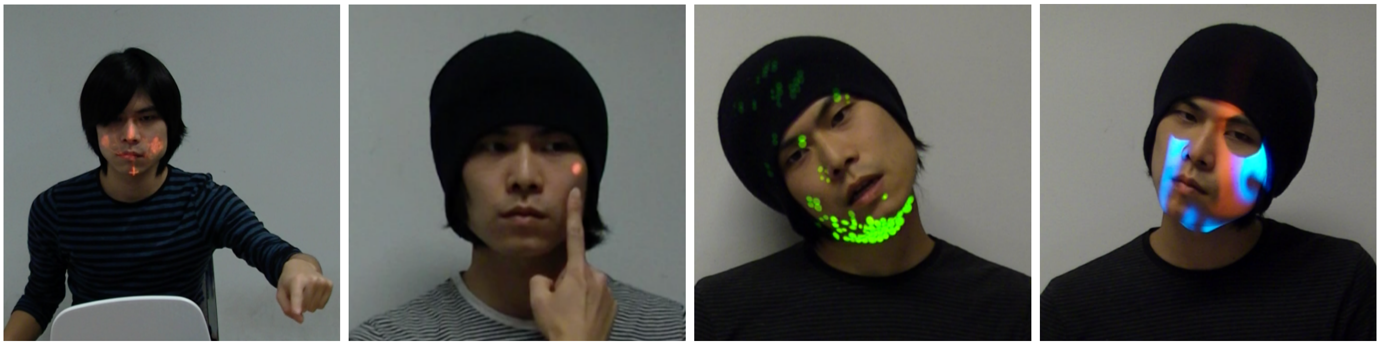
$$\|\mathbf{m_i} - \mathbf{m_i}'\|^2 = \|\mathbf{K}\ \mathbf{E}\ \mathbf{M_i} - \mathbf{m_i}'\|^2 \tag{2}$$

where $\mathbf{m_i}'$ is the projector pixel that has the same codeword as a depth camera pixel that corresponds to $\mathbf{M_i}$. Given a set of points, the fitting problem is to minimize the summation

$$\sum_i \|\mathbf{K}\ \mathbf{E}\ \mathbf{M_i} - \mathbf{m_i}'\|^2. \tag{3}$$

B. *Depth Camera Marker Tracking*

Marker tracking is often achieved by using two or more calibrated cameras, for example, a motion capture system; however, it requires multiple cameras and calibration, which



(a) Leap Motion hand tracking.

(b) A red sphere rendered close to the fingertip.

(c) Rigid-body simulation.

(d) Fluid simulation for virtual tears and blood.

Fig. 5. Photos of the improved face projection system. In Figure 5a and 5b, finger tracking methods are integrated to possibly assist drawing on the own face. In Figure 5c and 5d, 2D physics engines are applied to render virtual objects.

may not be feasible for demonstrations. Instead of a stereo camera setup, we use a depth camera to track a rigid-body object, combining IR and depth images. Since the LED markers of the object appear as blobs in an IR intensity image of the depth camera, these markers can be extracted by the OpenCV blob tracker, which not only returns the 2D blob coordinates (x, y) but also labels the blobs based on the previous coordinates. 2D coordinates of the LEDs can be located in the IR image since the LED emission intensity is greater than IR patterns projected by the depth camera; on the other hand, the depth values around the LEDs cannot be measured. To locate the 3D positions of the LED from the depth image, the depth values of surrounding pixels are averaged, assuming the region surrounding an LED is planar. Our implementation samples the four corners of the bounding box of the blob, and the average depth value is assigned as a the marker depth. Finally, the 2D coordinates and depth values are converted to 3D world coordinates using the focal length of the depth camera. These 3D coordinates are smoothed by Kalman filtering. Then, the pose of the rigid-body object, i.e., the rotation matrix and translation vector, compared to the initial pose is estimated by least-squares fitting [18] so that the face mesh can be transformed to the current position and orientation. The rotation matrix and translation vector are smoothed by the Kalman filter, and their velocities are used for a prediction filter as explained in the following section.

C. Prediction Filter

Various sources contribute to the overall system latency. These include the Kinect sensor framerate (33 ms at 30 Hz), the OpenGL refresh rate (17 ms at 60 Hz), the minimum projector latency (also 17 ms at 60 Hz), processing delay, communication time, USB overhead, and operating system scheduling. We used a 4-wheel robot with LED markers to measure the latency. The robot drove at constant velocity, and the marker tracking method in Section III-B was used to project a line between two markers, aligned perpendicular to the direction of motion. By measuring the distance between the physical markers and the projected line, latency was determined to be approximately 150 ms.

Fortunately, knowledge of the target object velocity and the system latency can be used to estimate the actual target position using the same prediction step of the Kalman filter, thereby compensating for this latency.

The position estimate is obtained by naively assuming a constant velocity

$$p_{k+\Delta t} = p_k + v_k \ \Delta t, \tag{4}$$

and the prediction filter is applied to the Euler angles as well

$$\mathbf{\Theta}_{k+\Delta t} = \mathbf{\Theta}_k + \mathbf{\Omega}_k \ \Delta t, \tag{5}$$

where $\mathbf{\Theta}$ and $\mathbf{\Omega}$ are vectors of Euler angles and angular velocities, respectively.

For periods of movement at constant velocity, experimental measurements indicated an impressive reduction in the average projected error from 42 pixels to 8 pixels, measured in the direction of user motion.

D. User Interaction

Participants wear the goggles containing IR LEDs. A small transparent region in front of the eyes allows the participants to view themselves in a mirror. A static picture of the participant with the goggles is captured by the depth camera during the 3D scanning (Figure 3) and displayed on a tablet interface. Using the tablet, other individuals can virtually draw on the participant's face, choosing either static or dynamic pen colors. Additional predefined graphical stickers, such as filled circles and manga-style eyes, can be added to the drawing.

The coordinates of the pen and stickers are sent to a Node.js server running on a main PC through a Websocket. Then, the Node.js server proxies the data by Open Sound Control (OSC) messages [21] to the renderer, which reproduces the drawing on the tablet.

IV. Enhancements for Effective Interactive Entertainment

The system was tested by over 100 attendees at an international virtual reality venue (Figure 4), from which our observations of user behavior led to the formulation of various improvements.

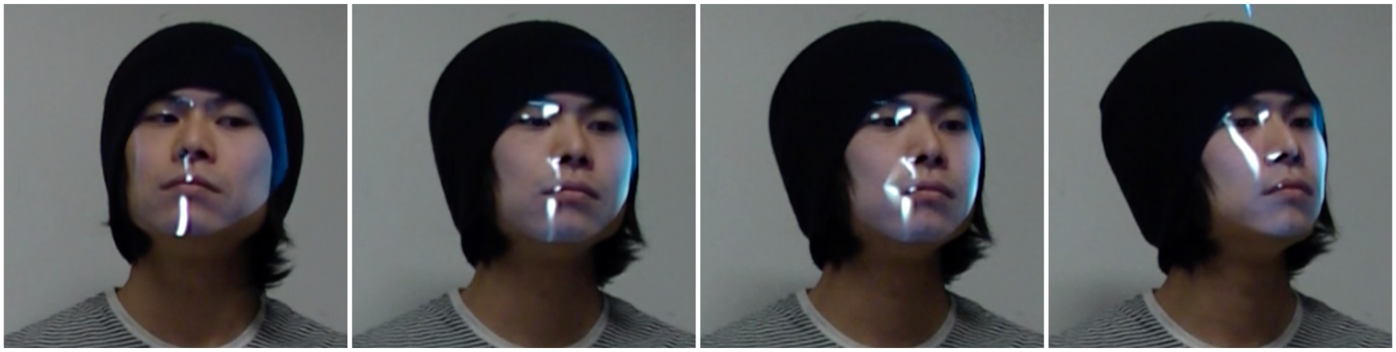


Fig. 6. Specular reflection shading. Highlights can be seen on the cheek and jaw, and move across the face according to the head orientation.

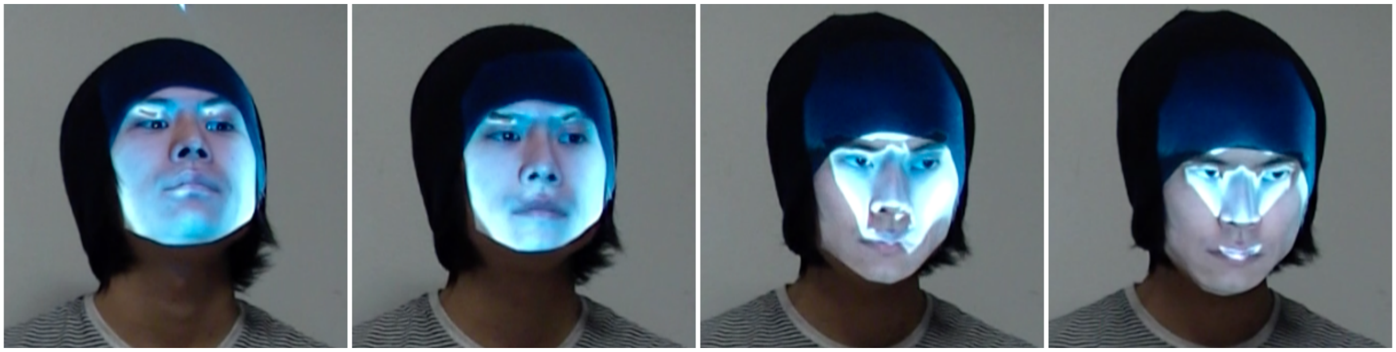


Fig. 7. Metal reflection shading. Similar to the specular reflection example, surface areas whose normals are parallel to the incident direction of light, i.e., similar to the optical axis of the camera that captured the photos above, are highlighted.

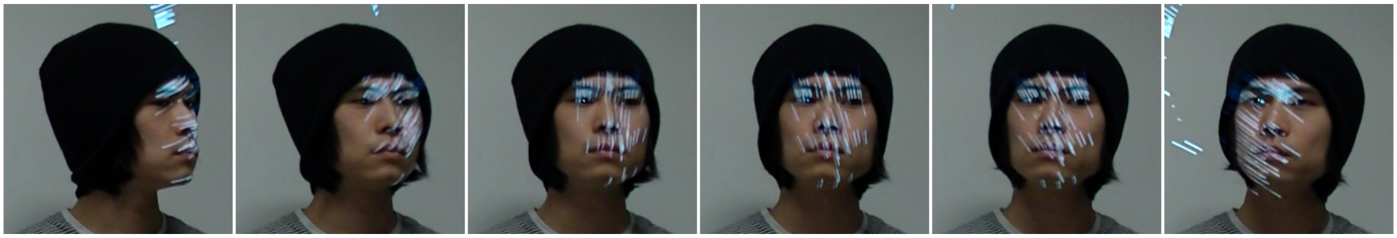


Fig. 8. Parallax rendering. The projected lines are visible through a mirror, and can be interpreted as parallel line segments extending from the face.

First, drawing on the tablet device can be replaced by aerial gestures or tracing on the face by a physical brush. For this purpose, the enhanced system makes use of a hand-tracking sensor, the Leap Motion device. This frees the participants from having to focus their visual attention on the tablet interface, allowing them to see changes in the projection itself, instantly, while drawing. In addition, the individuals whose faces are augmented can also benefit from this capability, viewing themselves in the mirror while drawing (Fig. 5a). Doing so with the tablet interface was challenging at best.

The coordinates of the user's index fingertip are extracted and linearly mapped to the texture coordinate ($0 \leq x < 1024$, $0 \leq y < 768$) of the face projection. To give feedback to the user, a cross-hair cursor is rendered at the corresponding current position on the texture. Although the finger path is continuously connected by a single line, we vary its alpha according to the velocity of the hand. Rapid hand motions, for example, while the user is searching for the cursor, result in a mostly transparent line, whereas slow drawing gestures result in more opaque lines.

Second, the requirement to wear goggles, which hide the user's eyes, was seen as a significant shortcoming. Fortunately, if the face is not occluded, e.g., by an accessory such as goggles, effective face tracking is possible directly by the Kinect for Windows SDK [5], without requiring the use of any optical markers. The face-tracking SDK generates a 3D face mesh, whose vertices are sent to our rendering application through Open Sound Control (OSC) messages [21]. The renderer then maps a texture on the mesh (Figure 9), setting the OpenGL projection matrix and modelview matrix according to the projector intrinsics and extrinsics, respectively. Finally, the rendered result is projected on the physical face.

Third, 2D physics engines are integrated so that the individual can interact with virtual objects on the face. One example uses a 2D physics engine [22], and a circle rigid-body polygon is generated every two frames around the forehead (Figure 5c). On the edge of the jaw, a virtual edge is defined

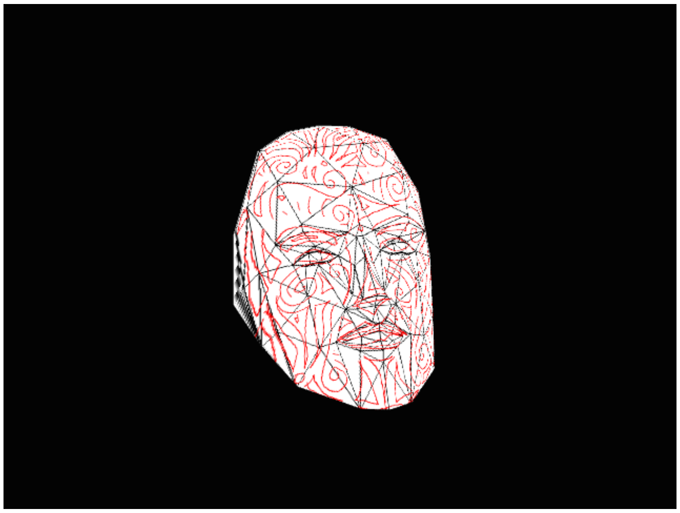


Fig. 9. An example of a texture-mapped mesh with overlaid wireframe.

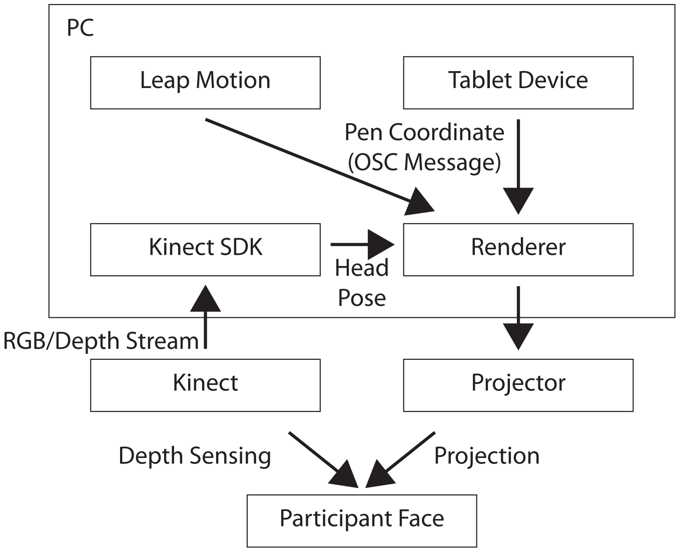


Fig. 10. A flowchart of the system.

which bounces polygons. The physics is simulated on a 2D plane that is mapped to the face mesh. Although the 2D plane is independent of the world coordinate system of the depth camera, the gravitational force of the physics engine is rotated in real time to correspond to the world. That is to say, when a head is rotated θ degrees about z axis, which is the axis the nose is pointing, the gravity is rotated $-\theta$ degrees. Thus, the participant can tilt the head to drop the polygons from the jaw. Note that two black circles are overlaid on the eye positions of the final texture to avoid projection on eyes.

Another example, a GPU-based 2D fluid simulator [23] is integrated. In Figure 5d, blue and red fluid is generated on the eyes and forehead to map virtual tears and blood, respectively. As in the previous example, the gravitational force is mapped to the world coordinate system. In this example, a virtual obstacle is defined on the nose instead of the jaw, so that fluid can branch at the nose.

Fourth, a primitive example of parallax rendering is tested, by defining a virtual line segment from the face vertices towards the facing direction. As shown in Figure 8, the lines, which are projected on the face, rotate to follow the head. From a viewpoint close to the projector, the lines can be seen with a parallax effect. By suitable placement of a mirror, the participant can also enjoy the effect, and can control the orientation of the line by rotating their head. Through an optical illusion, the lines can be thought as virtual line segments fixed to the face, and with appropriate curvature, a virtual "hairy face" can be simulated.

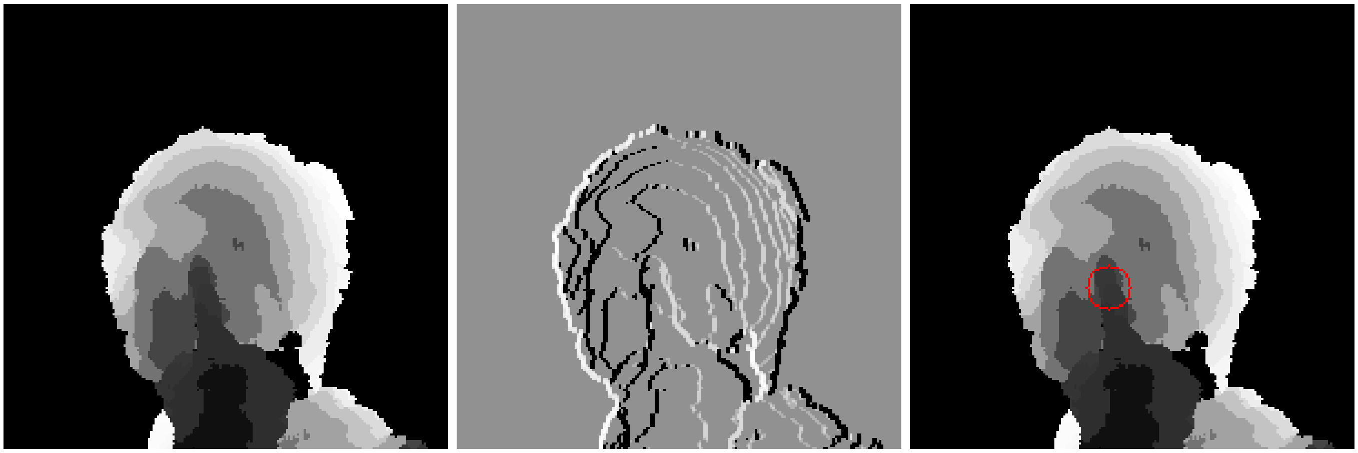
Fifth, relighting is demonstrated using Unity3D built-in shaders. Since the face geometry is known, a virtual light source and virtual face material can be defined to simulate the reflection. In one example, the projected face texture simulates specular reflection highlights on the cheek and jaw (Figures 6 and 7). This is achieved by placing a virtual, directional light source close to the sensor. Another example shows metallic reflection by physically based shading.

Finally, a simple fingertip detection algorithm is implemented. The algorithm is a simplified version of the method by Harrison et al. [24]. First, the skeletal joint of the head tracked by the Kinect for Windows SDK is projected on a depth map, and a 200×200 pixel region of interest surrounds the projected position (Figure 11a). A Sobel map is calculated along the x-axis (Figure 11b), as appropriate to identify a vertically oriented finger. Next, the algorithm searches along the x-axis for a convex structure of sufficient size, identified as an array of 30 pixels that transition from negative to zero to positive values. If such a convex structure continues along the y-axis, i.e., corresponding to the edges of the finger, the pixels found are labeled as a finger, and the top of the continuous structure is extracted as a fingertip (Figure 11c). In Figure 5b, a detected fingertip is rendered as a red sphere, and projected on the face.

Although such fingertip detection enables a natural interface for facial augmentation, the detection accuracy is limited by the depth resolution of the depth camera. Using commodity hardware, the difference between fingers and the face surface is often not detected. In such cases, both "touch detection" as well as finger detection itself, are not sufficiently robust for this approach to projected face drawing. Furthermore, given the convex structure of the nose, it is sometimes incorrectly detected as a fingertip. More problematically, fingers placed on the face can disrupt the face-tracking algorithm. With the current setup, it is therefore helpful to ensure that the head is stable during interactions when the face is partially occluded.

A flowchart of the tracking, projection and user interaction is shown in Figure 10.

More recently, the proposed system was presented at Laval Virtual 2015, a public exhibition of virtual reality and augmented reality technologies (Figure 12). Among the effects proposed above, fluid simulation was combined with Leap Motion hand tracking to support the ability to draw virtually on one's own face. This setup proved successful in that most participants were able to recognize the correspondence between their finger motion and the fluid simulation on the face.



(a) A depth map of a face and a hand.

(b) A Sobel map.

(c) A detected fingertip.

Fig. 11. Fingertip detection.

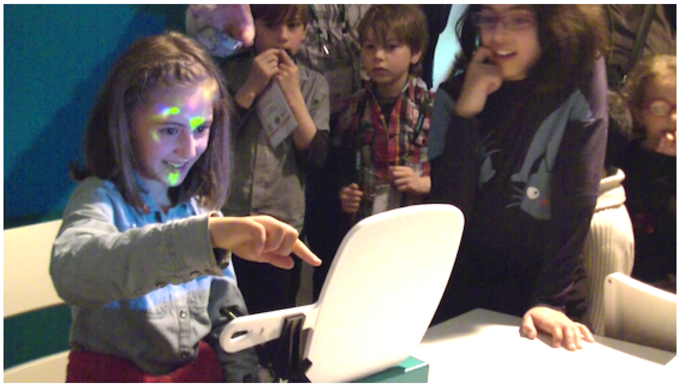


Fig. 12. SharedFace during a demonstration at Laval Virtual 2015.

V. Conclusions

We presented an interactive projection system that maps drawing and virtual objects on an individual's face. Given user feedback on the first prototype of facial projection mapping with markers, we designed a markerless face projection system. With the second prototype, participants do not need to wear devices in order to experience the system. The initial system required use of a pen display device for drawing on a face. We then integrated a hand-tracking sensor to improve the interaction experience for the individuals whose faces are "drawn upon", allowing them to draw on their own faces while viewing the results in a mirror. For this purpose, we implemented a simple fingertip-detection algorithm; however, the limitations of commodity depth sensors and our simplified assumptions of finger geometry limit the robustness of this approach.

To illustrate other application examples, we employed 2D physics engines for rigid-body and fluid simulation. These allow the individuals to tilt their heads to control virtual objects on the face. We also implemented re-lighting shaders to simulate virtual skin materials such that facial highlights change according to head position and orientation. Finally, we described a primitive parallax rendering example to demonstrate the possibility of integrating 3D rendering. More sophisticated augmented reality applications can be built with 3D bullet physics simulation exploiting a face 3D geometry to interact with virtual objects as well as face re-lighting and parallax effects from the previous examples.

Although we explored several animation effects for the SharedFace system, we have not yet studied human perception and reaction to such effects. For entertainment purposes, the system can be extended to support multiple participants, for example, allowing two individuals, facing one another or sharing a mirror, to draw on each other's face simultaneously.

Acknowledgments

We gratefully thank Dr. Yuki Hashimoto of the University of Tsukuba for granting us the use of his equipment for the first prototype, Dr. Akihiko Shirai of the Kanagawa Institute of Technology for valuable advice, and Mr. Daniel Biléu for designing the tattoo-like pattern.

References

[1] Chase No Face / BELL http://vimeo.com/26649425.

[2] Omote: Real-time Face Tracking and Projection Mapping http://vimeo.com/103425574.

[3] R. Raskar, J. van Baar, P. Beardsley, T. Willwacher, S. Rao, and C. Forlines, "ilamps: Geometrically aware and self-configuring projectors," in *ACM SIGGRAPH 2006 Courses*, ser. SIGGRAPH '06. New York, NY, USA: ACM, 2006. [Online]. Available: http://doi.acm.org/10.1145/1185657.1185802

[4] J. Scheible and T. Ojala, "Mobispray: Mobile phone as virtual spray can for painting big anytime anywhere on anything," in *ACM SIGGRAPH 2009 Art Gallery*, ser. SIGGRAPH '09. New York, NY, USA: ACM, 2009, pp. 5:1–5:10. [Online]. Available: http://doi.acm.org/10.1145/1667265.1667271

[5] Q. Cai, D. Gallup, C. Zhang, and Z. Zhang, "3d deformable face tracking with a commodity depth camera," in *Computer Vision ECCV 2010*, ser. Lecture Notes in Computer Science, K. Daniilidis, P. Maragos, and N. Paragios, Eds. Springer Berlin Heidelberg, 2010, vol. 6313, pp. 229–242. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-15558-1_17

[6] J. Salvi, J. Pags, and J. Batlle, “Pattern codification strategies in structured light systems,” *Pattern Recognition*, vol. 37, no. 4, pp. 827 – 849, 2004, agent Based Computer Vision. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320303003303

[7] C. Choi and H. Christensen, “Rgb-d object tracking: A particle filter approach on gpu,” in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, Nov 2013, pp. 1084–1091.

[8] S. Audet, M. Okutomi, and M. Tanaka, “Augmenting moving planar surfaces robustly with video projection and direct image alignment,” *Virtual Reality*, vol. 17, no. 2, pp. 157–168, 2013. [Online]. Available: http://dx.doi.org/10.1007/s10055-012-0210-9

[9] J. C. Lee, P. H. Dietz, D. Maynes-Aminzade, R. Raskar, and S. E. Hudson, “Automatic projector calibration with embedded light sensors,” in *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST ’04. New York, NY, USA: ACM, 2004, pp. 123–126. [Online]. Available: http://doi.acm.org/10.1145/1029632.1029653

[10] J. Lee, S. Hudson, and P. Dietz, “Hybrid infrared and visible light projection for location tracking,” in *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST ’07. New York, NY, USA: ACM, 2007, pp. 57–60. [Online]. Available: http://doi.acm.org/10.1145/1294211.1294222

[11] I. Garcia-Dorado and J. Cooperstock, “Fully automatic multi-projector calibration with an uncalibrated camera,” in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, June 2011, pp. 29–36.

[12] N. Hashimoto, K. Honda, M. Sato, and M. Sato, “Making an immersive projection environment with our living room,” in *Proceedings of the 8th International Conference on Virtual Reality Continuum and Its Applications in Industry*, ser. VRCAI ’09. New York, NY, USA: ACM, 2009, pp. 83–87. [Online]. Available: http://doi.acm.org/10.1145/1670252.1670271

[13] B. R. Jones, H. Benko, E. Ofek, and A. D. Wilson, “Illumiroom: Peripheral projected illusions for interactive experiences,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI ’13. New York, NY, USA: ACM, 2013, pp. 869–878. [Online]. Available: http://doi.acm.org/10.1145/2470654.2466112

[14] P. Lucey, J. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews, “The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression,” in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, June 2010, pp. 94–101.

[15] K. Fujii, M. Grossberg, and S. Nayar, “A projector-camera system with real-time photometric adaptation for dynamic environments,” in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, June 2005, pp. 814–821 vol. 1.

[16] Human Face Video Mapping by Oskar and Gaspar http://vimeo.com/39697056.

[17] P. Bourke, “Efficient triangulation algorithm suitable for terrain modelling,” in *Proc. Pan Pacific Computer Conf*, 1989.

[18] K. Arun, T. Huang, and S. Blostein, “Least-squares fitting of two 3-d point sets,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-9, no. 5, pp. 698–700, Sept 1987.

[19] S. Audet and M. Okutomi, “A user-friendly method to geometrically calibrate projector-camera systems,” in *Computer Vision and Pattern Recognition Workshops, 2009. CVPR Workshops 2009. IEEE Computer Society Conference on*, June 2009, pp. 47–54.

[20] Camera Calibration and 3D Reconstruction http://docs.opencv.org/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html.

[21] Opensoundcontrol.org http://opensoundcontrol.org.

[22] Box2D — A 2D Physics Engine for Games box2d.org.

[23] GPU Gems - Chapter 38. Fast Fluid Dynamics Simulation on the GPU http://http.developer.nvidia.com/GPUGems/gpugems_ch38.html.

[24] C. Harrison, H. Benko, and A. D. Wilson, “Omnitouch: Wearable multitouch interaction everywhere,” in *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST ’11. New York, NY, USA: ACM, 2011, pp. 441–450. [Online]. Available: http://doi.acm.org/10.1145/2047196.2047255