Mapping an Intel® RealSense™ SDK Face Scan to a 3D Head Model

This code sample allows the user to scan their face using a front-facing Intel® RealSense™ camera, project it onto a customizable head mesh, and apply post-processing effects on it. It’s an extension of a previous code sample titled Applying Intel® RealSense™ SDK Face Scans to a 3D Mesh. This sample adds features and techniques to improve the quality of the final head result as well as provide a series of post mapping effects. The code sample is available on GitHub here.

The following features have been added in this code sample:

- **Parameterized head.** The user can shape the head using a series of sliders. For example, the user can change the width of the head, ears, jaw, and so on.
- **Standardized mesh topology.** The mapping algorithm can be applied to a head mesh with a standardized topology and can retain the context of each vertex after the mapping is complete. This paves the way for animating the head, improved blending, and post-mapping effects.
- **Color and shape post processing.** Morph targets and additional color blending can be applied after the mapping stages are complete to customize the final result.
- **Hair.** There are new hair models created to fit the base head model. A custom algorithm is used to adjust the hair geometry to conform to the user’s chosen head shape.

The final results of this sample could be compelling to various applications. Developers could use these tools in their applications to allow their users to scan their face and then customize their character. It could also be used to generate character content for games. Additional morph targets could be added to increase the range of characters that can be created.

![Figure 1](image.png)

**Figure 1:** From left to right: (a) the face mesh returned from the Intel® RealSense™ SDK’s scan module, (b) the scanned face mapped onto a head model, (c) the head-model geometry morphed with an ogre morph target, and (d) the morphed head colorized.
Using the Sample
The sample includes two executables: ReleaseDX.exe, which supports face scanning, and ReleaseDX_NORS.exe, which only supports mapping of a previously scanned face. Both executables require the 64-bit Visual Studio* 2013 Runtime which can be downloaded here. ReleaseDX.exe requires the installation of Intel® RealSense™ SDK Runtime 2016 R1 (8.0.24.6528), which can be downloaded here or here.

Running the ReleaseDX.exe will begin the face-scanning process. Once the head-positioning hints are satisfied, press the Begin Scan button, turn your head slowly from side to side, and then click End Scan. For best results remove hats or glasses, pull back your hair, and scan in a well-lit area.

Once the scan is complete, or an existing scan has been loaded, the following categories in the UI can be used to customize the result:

- **Face scan.** Adjust the yaw, pitch, roll, and z displacement to adjust the face scan’s orientation and position.
• **Head shaping.** Use the provided sliders to change the shape of the head model. The idea is to build a head that matches the head of the person scanned. Adjustments to this shape can be made in the post-processing stage.

• **Blending.** Use the color controls to choose two different colors that best match your skin color. The first color is the base color and the second color is for tone.

• **Post head shaping.** Make any shape adjustments to the head that you want performed after the mapping process. In this stage you can do things like change your body mass index, turn yourself into an ogre, make your ears big, and so on.

• **Post blending.** Select any color effects to apply on the entire head after the mapping is complete. These color effects won’t affect the lips or eyes. These effects will let you adjust or colorize the hue/saturation/luminance of the head.

The debug category contains many options for visualizing different parts of the face-mapping pipeline.

The sample allows exporting the resulting head and hair to an .OBJ file so that it can be loaded into other applications.

![Figure 3: A screenshot from the code sample showing a small subset of the options available for customizing the head.](image)

**Art Assets**

The art assets used in this code sample are briefly described in this section and will be referenced throughout the article. All texture assets are authored to the UV coordinates of the base head model. All texture assets are used during the blending stage of the pipeline with the exception of the displacement control map, which is used in the geometry stage.

• **Base head mesh.** Base head mesh on which the scanned face mesh will be applied.
- **Head landmark data.** Landmarks on the base head mesh that coincide with the landmarks the Intel RealSense SDK provides with each face scan.
- **Displacement control map.** Controls which vertices of the base head mesh are displaced by the generated face displacement map.
- **Color control map.** Controls blending between the face color and the head color.
- **Feature map.** Grayscale map that gives the head texture for the final generated diffuse map.
- **Skin map.** Used in the post blending stage to prevent color effects from affecting the eyes and lips.
- **Color transfer map.** Controls the blending between the two user-selected colors.
- **Landmark mesh.** Used to shift head vertices to their corresponding locations on the face map projection.
- **Head morph targets.** Collection of morph targets that can be applied to the base head shape both before and after the face is projected onto the head.
- **Hair models.** Collection of hair models that the user can select between.

### Face-Mapping Pipeline

The face-mapping pipeline can be separated into four stages.

1. **Displacement and color map.** We render the face scan geometry to create a displacement and color map that we project onto the head in a later stage.
2. **Geometry.** This stage morphs the positions and normals of the head mesh. The values from the displacement map are used to protrude the face scan’s shape from the head model.
3. **Blending.** We blend the head and face color channels together and output a single color map that maps to the UVs of the head model.
4. **Hair geometry.** We remap the hair vertex positions to account for changes in the head shape made during the geometry stage.

### Face Displacement and Color Map Stage

During this stage, the scanned face mesh is rendered using an orthographic projection matrix generating a depth map and color map. The face scan landmark positions are projected onto these maps and will be used in the geometry stage to project onto the head mesh.
Geometry Stage

This stage morphs the base head mesh shape and imprints the displacement map onto the head. This must be done while maintaining the context of each face vertex; a vertex on the tip of the nose of the head mesh will be moved so that when the face displaces the vertices, that vertex is still on the tip of the nose. Maintaining this context allows for rigging information on the base head mesh to persist on the resulting head mesh.

Details on this process are outlined in the sections below. The high-level steps include the following:

1. Project the base head mesh vertices onto the landmark mesh. This will associate each head vertex with a single triangle on the landmark mesh, barycentric coordinates, and an offset along that triangle's normal.
2. Morph the head vertices using morph targets.
3. Compute a projection matrix that projects the 2D displacement/color maps onto the head mesh and use it to calculate the texture coordinates of each vertex.
4. Morph the landmark mesh using the face landmark data.
5. Use the projection data from step one to shift the head vertices based on the morphed landmark mesh.
6. Displace the head vertices along the z-axis using the displacement map value.
7. Apply post-processing morph targets.

Building a Parametric Head

A wide range of head shapes are available for the morphing system. Each shape sculpts a subset of the head’s range (for example, one of the targets simply controls the width of the chin. Another, body-mass-index (BMI), changes almost the entire head shape.) Each artist-authored head shape contains the same number of vertices, which must match the corresponding vertices in the base head shape.
The artist-authored head shapes are turned into morph targets by compiling a list of delta positions for each vertex. The delta position is the difference, or change, between the vertex of the base head mesh and its associated target shape vertex. Applying a morph target is done by adding the delta positions for each vertex multiplied by some scalar value. A scalar of zero has no effect, while a scalar of one applies the exact target shape. Scalars above one can exaggerate the target shape, and negative scalars can invert it producing interesting effects.

The sample exposes some compound morph targets that allow a single slider to apply multiple morph targets. Some of the sliders apply a weight of zero to one, while others might allow values outside this range.

These morphing techniques can be applied both before and after the face is mapped to the head mesh.

**Creating the Displacement and Color Map Projection Matrix**

The displacement/color map projection orthographically projects head model vertex positions into UV coordinates of the Displacement and Color maps that were previously created. For more details on this process see the documentation of the previous sample.

**Fitting-Face Geometry**

The previous sample required the base head mesh to have a relatively dense vertex grid for the facial area. It displaced these vertices to match the scanned mesh’s shape. However, it didn’t discern vertices by their function (for example, vertex in the corner of the mouth). In this version of the sample the base head mesh is less dense and the vertices are fit to the face scan, preserving vertex functionality. For example, the vertices around the eyes move to where the scan’s eyes are.

The Intel RealSense SDK provides a known set of face landmarks with the face scan. The authored base head mesh supplies matching, authored landmarks. A landmark mesh is used to map between faces. It has one vertex for each important landmark, forming relatively large triangles that subdivide the face. We identify where each base head mesh vertex projects onto the landmark mesh to compute its corresponding position on the scanned face.

During this process, the head vertices are projected onto the landmark mesh, the landmark mesh is morphed based on the base head mesh and face landmark data, and the vertex positions are reprojected onto the head. Finally, the displacement map is applied to the z component of each facial area vertex to extrude the scanned face shape. The displacement control map ensures only the face vertices are shifted and that there is a smooth gradient between vertices that are and are not affected.
The projection of vertices onto the landmark mesh is similar to the projection done in the hair-fitting stage.

**Figure 6:** Face color map with landmarks visible (left) and authored base head mesh with head landmarks visible (right).

**Figure 7:** The landmark mesh overlayed on top of the head mesh (left). Note that the inner vertices all line up over head landmarks. The landmark mesh morphed based on the face landmark information (right).
Figure 8: The head with landmark mesh overlay after reprojecting vertex positions and displacing them (left). Notice how the lips have been shifted upward. The head after reprojecting, displacing, and applying the face color map (right).

**Blending Stage**

The blending stage blends together the face scan color data and the artist-authored head textures producing a final diffuse texture built for the head UV coordinates.

The color transfer map is used to lerp between the two user-selected head colors. That resulting color is then multiplied by the head detail map to create a final color for the head. This color control map is then used to blend between that head color and the face color creating a smooth transition between.

After the color is determined, we can optionally apply some post-processing blending effects. The sample supports a colorize and color adjust effect. The colorize effect extracts the luminosity of the final blended color and then applies a user-specified hue, saturation, and additional luminance to it. The color adjust is similar except it adjusts the existing hue, saturation, and luminance instead of overriding them. Both of these effects support two colors/adjustments that are controlled by the color control map. These effects use the skin map mask allowing the color of the eyes and lips to remain unchanged.

All of this blending is done on the GPU. The shader can be found in Media/MyAssets/Shader/SculptFace.fx.
Figure 9: The Color Transfer Map, which is used to blend between the two user selected colors (left), and the Feature Map, which adds texture to the head (right).

Figure 10: The Color Control Map (left) controls blending between the head color and the face color. The blending process creates a color texture that maps to the head mesh’s UV coordinates (right).

Hair Geometry Stage
Hair information isn’t available from a face-only scan. Producing a complete head requires the application to provide hair. This sample includes only a few choices, with the intent being to demonstrate the technical capability and not to provide a complete solution. A full-featured application could include many choices and variations.
The sample supports changing the shape of the head, so it also supports changing the shape of the hair to match. One possibility might be to create hair morph targets to match each head morph target. However, this would create an unreasonable amount of required art assets, so the sample instead programmatically adjusts the hair shape as the head shape changes.

Figure 11: The base hair rendered on the base head mesh (left). Fitted morphed hair rendered on morphed head shapes (center and right).

The hair fitting is accomplished by locating each of the hair vertices relative to the base head model, then moving the vertices to the same head relative positions on the final head shape. Specifically, each hair vertex is associated with a triangle on the base head mesh, barycentric coordinates, and an offset along the normal.

An initial approach to mapping a hair vertex to a triangle on the base head mesh would be to iterate over each head mesh triangle, project the vertex onto the triangle’s plane using the triangle’s normal, and check whether the vertex lies within the triangle. This approach yields situations where a hair vertex might not map to any of the triangles. A better approach is to instead project each base head mesh triangle forward along each of the vertices’ normals until it intersects the hair vertex. In the event that a hair vertex can map to multiple head mesh triangles (since the head mesh is non-convex), the triangle closest to hair vertex along that triangle’s normal is chosen.
Figure 12: A simplified head mesh and hair vertex.
Figure 13: A single triangle is extruded along the vertex normals until it contains the hair vertex.
Figure 14: The new hair vertex position is calculated for the new head mesh shape. It's located by the same barycentric coordinates, but relative to the triangle’s new vertex positions and normals.
Figure 15: Projecting the triangle onto the vertex - the math.

Figure 15 shows the vectors used to project the triangle onto the vertex, as seen from four different views.

- **Yellow triangle** is the original head triangle.
- **Gray triangle** is the yellow triangle projected onto the hair vertex.
- **Blue lines** represent the vertex normals (not normalized).
- **Pink line** is from the hair vertex to a point on the yellow triangle (note that one of the vertices serves as a point on the plane).
- **Green line** shows the shortest distance from the vertex to the plane containing the triangle, along the triangle’s normal.

The projected triangle’s vertex positions are computed by first computing the closest distance, $d$, from the hair vertex to the plane containing the triangle.

$N_t = \text{Triangle normal}$

$N_s = \text{Surface normal (that is, interpolated from vertex normals)}$

$V_h = \text{Hair vertex}$

$P = \text{Position on the plane (that is, one of the yellow triangle’s vertices)}$

$P_x = \text{Intersection position. Position on the triangle intersected by the line from the hair vertex, along the surface normal}$

$a_i = 2\times \text{the projected triangle area}$
\[ d = \text{Closest distance from the hair vertex to the plane containing the head triangle} \]
\[ l = \text{distance from the hair vertex to the intersection point} \]

The closest distance from the hair vertex to the plane containing the triangle gives the distance to project the triangle.

\[ d = \mathbf{N}_i \cdot (\mathbf{V}_h - \mathbf{P}) \]

The position of each projected vertex \( v'_i \) where \( \mathbf{N}_i \) is normal for \( \mathbf{V}_i \), is

\[ v'_i = \mathbf{V}_i + \frac{d}{\mathbf{N}_i \cdot \mathbf{N}_i} \]

The barycentric coordinates of the hair vertex, relative to the projected triangle are a function of the total triangle area. Twice the triangle's area is computed with a cross product.

\[ a_i = \frac{\| (v'_{i1} - v'_{i0}) \times (v'_{i2} - v'_{i0}) \|}{a_t} \]

The three barycentric coordinates \( a, b, \) and \( c \) are then

\[ a = \frac{\| (v'_{i2} - v'_{i1}) \times (v_h - v'_{i1}) \|}{a_t} \]
\[ b = \frac{\| (v'_{i0} - v'_{i2}) \times (v_h - v'_{i2}) \|}{a_t} \]
\[ c = \frac{\| (v'_{i1} - v'_{i0}) \times (v_h - v'_{i0}) \|}{a_t} \]

The vertex lies inside the triangle if \( a, b, \) and \( c \) are greater than zero, and the sum of \( a, b, c \) are less than one. Note that this vertex lies on the surface normal line. The barycentric coordinates give the surface normal, interpolated from the three vertex normals. They similarly give the point on the triangle that also lies along that line (that is, the intersection point).

\[ \mathbf{N}_i = a \mathbf{N}_0 + b \mathbf{N}_1 + c \mathbf{N}_2 \]

The intersection point is

\[ \mathbf{P}_x = a \mathbf{V}_0 + b \mathbf{V}_1 + c \mathbf{V}_2 \]

The distance \( l \) is stored from the hair vertex to \( \mathbf{P}_x \). After the head is deformed, the hair vertex is moved this distance away from the triangle on the new head shape.

\[ l = \| \mathbf{P}_x - \mathbf{V}_h \| \]

The process is reversed to determine the hair vertex's position relative to the head's new shape. The intersection's barycentric coordinates (previously computed relative to the base head mesh) are used to compute the hair vertex's position and normal on the new head.

\[ \mathbf{N}'_i = a \mathbf{N}'_0 + b \mathbf{N}'_1 + c \mathbf{N}'_2 \]
\[ \mathbf{P}'_x = a \mathbf{V}'_0 + b \mathbf{V}'_1 + c \mathbf{V}'_2 \]

The new hair vertex position is then

\[ \mathbf{V}'_h = \mathbf{P}'_x + l \mathbf{N}'_i \]
Note that this approach just moves the vertices. It doesn't check for other intersections, and so on. It produces excellent results in practice, but it does have limits. For example, an extreme head shape can poke a head vertex/triangle(s) through the hair.

Also note that the hair vertices are allowed to be on either side of the head triangle. This supports the case where an artist pushes some hair vertices inside the head. This distance is clamped to minimize the chances of associated a vertex with a triangle on the other side of the head.

Acknowledgements

Assets for the base head mesh, morph targets, textures, and some of the hair models were created by 3DModelForge (http://3dmodelforge.com).

Additional hair was created by Liquid Development (http://www.liquiddevelopment.com/).

Doug McNabb, Jeff Williams, Dave Bookout, and Chris Kirkpatrick provided additional help for the sample.

Notices

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

This sample source code is released under the Intel Sample Source Code License Agreement.

Intel, the Intel logo, and Intel RealSense are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

© 2016 Intel Corporation.