Facial Feature Detection for Automatic Head Modeling

Vanna Bushong
University of California, Los Angeles
vbushong@ucla.edu

Abstract

This project presents a method for automatically generating 3D models of the human head using two images as input - a front view and a side view of the subject. Key facial landmarks such as the chin, corners of the eyes and mouth, and tips of the ears are identified on the images using a combination of trained classifiers, corner and edge detection, and constraints of human facial geometry. The landmark coordinates are then used to translate corresponding vertices of a generic head model to match the images. The goal of this project was to provide a tool for artists that would offer a head start in character modeling for animated films or games; therefore, the final program was developed as a plugin for Autodesk Maya software, the industry standard for modeling and animation.

1 Introduction

The human head is one of the most difficult tasks to tackle when it comes to character modeling. In the animation and gaming industry, the standard practice is to obtain two images of the subject's head (a front and side view) and load these reference images into Maya or another modeling program. Using orthographic views, the modeler can begin with a primitive cube and then make adjustments by adding and manipulating vertices and edges until the desired head shape is achieved. Typically, only one half of the head is modeled at first, then the geometry is mirrored to produce a perfectly symmetrical other half. To account for asymmetries, the modeler must make further adjustments.

For even an experienced modeler, this is a time-consuming process. This project aims to provide a tool that speeds up the procedure and creates a head model that is as close as possible to the reference images. While others have been successful in generating models based on images [1] [5], the end results would benefit from more user interaction throughout the process. This project detects facial feature landmarks in two given images and allows the user to make adjustments if desired before the model is generated. The feature points are loaded into a plugin for Maya, which uses the coordinates of these points to adjust the vertices of a generic head. Since the head is created in Maya, the modeler can easily make any touch-up adjustments and begin using the model immediately.

This paper outlines the approach used in detecting the 28 facial landmarks that are currently used to manipulate the model, along with the steps taken to adjust the vertices of the final model. The initial problem was to detect a face in the two reference images along with the location of each of the main features. Once the general location of the features was established, the key landmarks were set using a combination of corner and edge detection, as well as the constraints of human facial geometry. For detecting the edges of lips, red hues were also factored in. After allowing the user to make adjustments, the coordinates were saved and loaded into the final program, which converts pixel coordinates to world space locations in centimeters for Maya.

The resulting program was successful in creating models with features closely aligned to the reference images. The tool has not been perfected, but it is already proving to be helpful in giving modelers a head start on their work.

2 Basic Feature Detection

The first step in building the model was to obtain two reference images of a subject. One of these images must be taken from the front of the subject's face, while the other must be from the left side. For best results, the subject should be well lit and placed in front of a solid colored background if possible (though not absolutely necessary). Long hair should be pulled back so the ears will be visible, and accessories such as glasses and earrings should be removed.

The next step was to detect the face in each of the target images. This was accomplished using a function from the OpenCV library that implements the Viola-Jones algorithm [6] using a trained classifier for faces. The classifiers for front and profile faces were applied to the front and side view images respectively.

After finding the faces, classifiers for the eyes, nose, mouth, and ears were applied in order to find the approximate location for each of these features. Results of the initial detection are shown below.

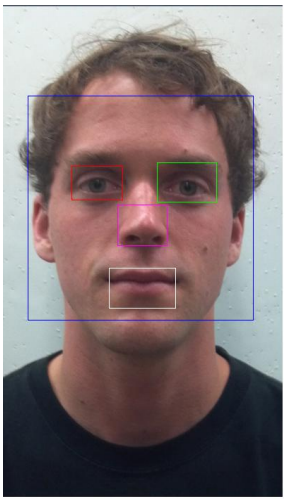

Figure 1: *Results of the initial feature detection on the front view of the subject.*

Figure 2: *Results of the initial feature detection on the side view of the subject.*

3 Eyes

In the current state of this project, four landmark points are set along the edges of each eye – at the top, bottom, and in both corners. The initial coordinates of these points were set by placing them halfway between the width and height of the detected eyes.

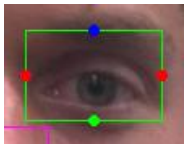

Figure 3: *Initial coordinates set for the left eye.*

A region of interest was created using the boundaries of the detected eyes and the resulting image was converted to grayscale and blurred using a 3x3 kernel. After this preprocessing, an OpenCV function for Canny edge detection was applied to the ROI. Canny edge detection works using a gradient calculation, followed by thresholding and non-maximum suppression of the image [2].

Figure 4: *Canny edge detection applied to the left eye.*

Using the Canny images, the top and bottom points of the eyes were snapped to the closest contour. To prevent error, the points were constrained from moving any farther than half the distance of the detected eye boundary, making overlap impossible.

For the corners, a new ROI was created using half the height of the detected eye and 25 percent of the width. This area was centered along the left and right edges of each eye, and a corner detector was applied. The detector was another function of OpenCV called goodFeaturesToTrack, which is an implementation of the Shi-Tomasi algorithm [4]. The function accepts a parameter for the number of corners to return, and returns them in order of strength. This parameter was set to one, thus returning only the strong corner that is found on the eye.

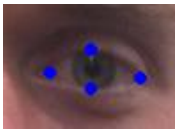

Figure 5: *Final coordinates set for the left eye after edge and corner detection.*

4 Nose

Defining the shape of the nose was a more difficult task. Three coordinates were initially set at the bottom, left, and right sides of the area detected by the nose classifier. This allowed for a rough definition of the outer edges of the nostrils and the point where the bottom of the nose meets the face.

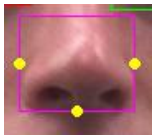

Figure 6: *Initial coordinates set for the nose.*

Just as with the mouth, Canny edge detection was applied to the ROI and the result was used to snap the points to the nearest contour. For the bottom point, only the bottom 25 percent of the ROI was searched, as it was discovered that if a contour was not found by that point, one did not exist. In poorly lit images, the edge at the bottom of the nose was not defined enough to be detected. If the entire nose was searched, the point would erroneously snap to a contour much too high. In this case, the point at its initialized position is a closer approximation than finding a contour. For the left and right side coordinates, the search for a contour began at each edge and covered half of the width of the detected nose region so the points could not overlap.

Figure 7: *Final nose coordinates after edge detection.*

The three coordinates in the front view gave a good starting shape for the nose, but one more point was needed to define the nose along the Z-axis. This point was defined as the tip of the nose in the side view image. To find this point, another ROI was created by extracting the left half of the detected face in the profile reference image. Once again, Canny edge detection was used to create an outline of the face. With the outline drawn, a search algorithm located the point on the contour with the smallest x component, making it the left-most point and the tip of the nose.

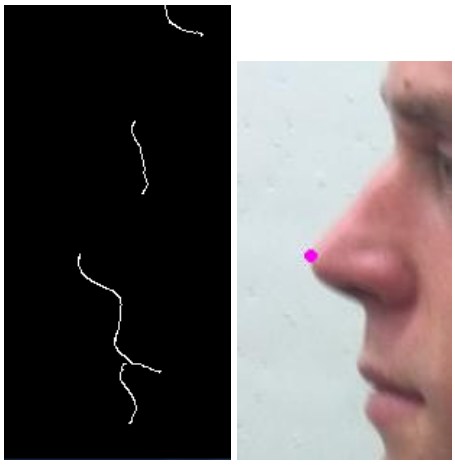

Figure 8: *Edge detection (on left) used to find the tip of the nose in the profile image (on right).*

This procedure makes it important to use a solid color background when photographing the subject, with the subject and lights positioned so as not to cast shadows on the background. This will prevent false positives from contours behind the face. However, as previously mentioned, the user interaction component of the program allows for mistakes to be corrected in case the modeler has no control over how the images are obtained.

5 Mouth

The edges of the mouth were detected using a slightly different method than the other features. The process began the same way as the eyes, with a point set halfway along each edge of the detected mouth region.

Figure 9: *Initial coordinates set for the mouth.*

Instead of detecting edges, however, the top and bottom points were adjusted by searching along the Y axis for a pixel falling within the color range of lips.

First, the RGB values were extracted from each pixel in the search. Then the brightness value Y was calculated in accordance to the ITU-R BT.601 standard for brightness in MPEG and JPEG algorithms:

$$Y = (0.299 * B) + (0.587 * G) + (0.114 * R) \quad (1)$$

The color components were then divided by Y and multiplied by 100 for scaling purposes.

$$C_r = R/Y * 100$$

$$C_g = G/Y * 100$$

$$C_b = B/Y * 100 \quad (2)$$

Nakata and Ando [3] have shown that the color most likely to represent lips will have a red component above 125, a green component between 80 and 95, and a brightness value below 180. Using these constraints, the top and bottom points were moved to a more accurate location.

After the top and bottom points of the mouth were set, it was easier to adjust the sides. The color detection process did not work in this case, since the corners of the mouth are usually much darker and in shadows. A new ROI was created for both the left and right sides, using the new top and bottom points of the mouth to set the height. The width of each was set to 25 percent of the width of the originally detected mouth. Just as with the eyes, the Shi-Tomasi corner detector was applied to find the left and right corners.

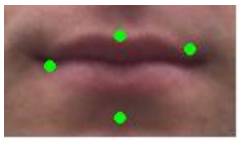

Figure 10: *Final coordinates for the mouth. In this example, the bottom and left points need adjustment by the user.*

6 Ears

Ears are probably the most difficult feature of the head to model, considering their intricate shape. In the current state of the project, the details of the ears (such as inner ear shape and ear lobe size) on the generic model are not adjusted for the final result. However, the generic ear is moved to the correct location on the final model using coordinates of the top and bottom of each ear in the reference images. With the ear in the correct position, the modeler should find it easy to make adjustments to the fine details.

For the front view, the ear points were established using knowledge of facial geometry and corner detection. Along the Y axis, most ears fall between the middle of the eyes and the bottom of the nose. Along the X axis, a person's head is generally equal to the width of five of his or her eyes. Using these guidelines, a region of interest was established at the top and bottom of each ear. The top region began at the top outside corner of each detected eye region and extended the same width and height as the eye region. The bottom region began at the same X axis location, but with a Y value taken from the top of the detected nose. After experimentation, the height of this region needed some room for error, since ear lobes can sometimes hit far above or below the bottom of the nose line. Therefore, the height of the bottom region was set to twice the height of the nose region.

With the ROIs established, the top and bottom points of the ear were detected using the Shi-Tomasi corner detector.

Figure 11: *The top region of the left ear. The white point is the detected corner.*

7 Chin and Head

Six additional points were necessary for establishing the size of the head. On the front image, one point was set at the bottom of the chin and another at the top of the head. These points would eventually be used to calculate the height of the head in pixels, which was then converted to centimeters for Maya.

Figure 12: *Coordinates of the chin and top of head as initially set. These will require user modification.*

These points were the most difficult to detect using contours or corner detectors. Therefore, the chin point was simply set halfway along the bottom of the face initially detected by the classifier. It was hard to know if this point was sitting above or below the actual chin, since the result from the classifier is not consistent one way or the other. It was also difficult to find the contour of the chin without picking up a lot of other contours created by shadows under the lips and on the neck. Because of this, the best location was generally found by leaving the point alone.

The top of the head was calculated using the knowledge that the middle of the eyes sit approximately halfway between the chin and the top of the head. In the future, a better method will be used to fine-tune the chin and head locations.

Two points were also established on the profile view for calculating the height of the head. An ROI was set for the chin over the bottom left corner of the detected face. After detecting edges and drawing the chin contour, the point for the bottom of the chin was set at the bottom right contour.

Figure 13: *Bottom of chin set using edge detection.*

The point at the top of the head was established the same way, with the ROI being set at the top center of the image. The same process was also used for finding a point along the front edge of the chin and the back of the head. These points were used to adjust the size of the head along the Z axis.

Figure 14: *Profile features set using edge detection.*

As previously noted, all of the points established using edge detection require a solid colored background for best results.

8 Setting up Maya

One of the key features of this project is the ability for the user to guide the modeling process as much or as little as desired. Once the facial landmarks have been identified, the user is allowed to adjust them by clicking and dragging the points on each image.

Figure 15: *Initially detected coordinates (on left) vs. coordinates after user adjustment (on right).*

Once he or she is satisfied, the coordinates are saved into a text file. To run the final plugin in Maya, the user specifies this file, along with the two reference images, to build the model. By default, Maya has orthographic camera views created for the

front and side of the workspace, so the two images are loaded into their respective image plane sitting perpendicular to each camera.

The default working unit in Maya is centimeters, and the height of the generic head model used is eight cm. To size the images properly, the height of the head in pixels was calculated by subtracting the top of the head from the bottom of the chin. The height was then divided by eight to find the number of pixels per cm in the image. Naturally, the inverse is the number of cm per pixel. This is a key value used in adjusting the model vertices, and must be calculated separately for each image.

$$heightInPx = chin_y - headTop_y$$

$$pxPerCm = heightInPx / 8$$

$$cmPerPx = 1 / pxPerCm \quad (3)$$

By multiplying the height of each image in pixels by their respective cmPerPx value, the images were scaled so that the top and bottom of the head align with the top and bottom of the model. Using this method, every head modeled will have the same height, but different widths. This is not a problem, since the final head will be placed on a character and scaled proportionally in x, y, and z to the appropriate size.

Figure 16: *Front and side images loaded into orthographic views in Maya.*

Once the images were sized, the generic head model was loaded and ready to be morphed into its new shape. Since each vertex in Maya has an index number, the numbers for the key vertices of the model corresponding to the 28 facial landmarks were identified ahead of time and loaded as constants into the program.

9 Adjustments along X, Y, and Z

The first adjustments made to the model were along the x and y axis, using the front image only. The generic head model used in this project contained 2179 vertices. Naturally, it was no simple task to move one vertex to a new location while simultaneously adjusting the surrounding vertices to create the proper head shape. The solution to this problem was to use Maya's soft select tool.

Soft select works by selecting a vertex plus all the surrounding vertices within a certain distance known as the falloff radius. The vertices within the falloff radius are weighted from zero to one, with the strongest weight being closest to the selection. After some trial and error, the falloff radius was set for every key vertex so that each move did not make an undesired impact on other key vertices.

One drawback of using the soft select tool was that vertices could not be translated by calling the Maya command to move to an absolute position in world space. If the selected vertex was moved to a specific position, all of the vertices included in the soft selection would move towards that point. The desired behavior was to move all of the selected vertices in the same direction. To accomplish this, it was necessary to use the Maya command for moving a vertex relative to its current position.

Calculating the relative change in position required finding the difference in centimeters between the where you want the vertex to be (the landmark point) and where you are now (the corresponding vertex on the model). The position of the vertex on the model was easy to obtain through a Maya function call, which returned the location in world space in centimeters. For the landmark position, all that was initially known was the x and y location on the image. To find a position on the x axis, the x location of the chin point on the front image was considered to be at the origin. Subtracting the x value of the chin from the x value of the landmark gave the offset in pixels. This value was then multiplied by cmPerPx to convert the result to centimeters.

$$abs\_landmark_x = (landmark_x - chin_x) * cmPerPx \quad (4)$$

With the two absolute locations known, the relative movement was simply the difference between them.

$$\Delta x = abs\_landmark_x - abs\_vertex_x \quad (5)$$

The same procedure was applied for calculating y values, using the chin as y at 0.

Adjustments along the z axis were made using the points on the profile image. A similar process was used as with the x and y axis, except in this case, the x values on the image correspond to z values on the side view. The difference was multiplied by cmPerPx for the side image to find the absolute location of the landmark.

10 Results

The resulting program creates a head with features that match fairly well to the reference images. While it is not perfect, the tool is certainly useable and brings the modeler much closer to a finished product than starting from scratch, which was the goal of the project.

The resulting images below show the models created by the program with no user modification in Maya. It is obvious that the program could be improved by adding more feature points. In the future, points will be added to the eyebrows, inner edges of the lips, the jaw line, and other locations along the edge of the head. Also, there have been no adjustments made to the neck or the shape of the ears - something that will be done in future versions of the program.

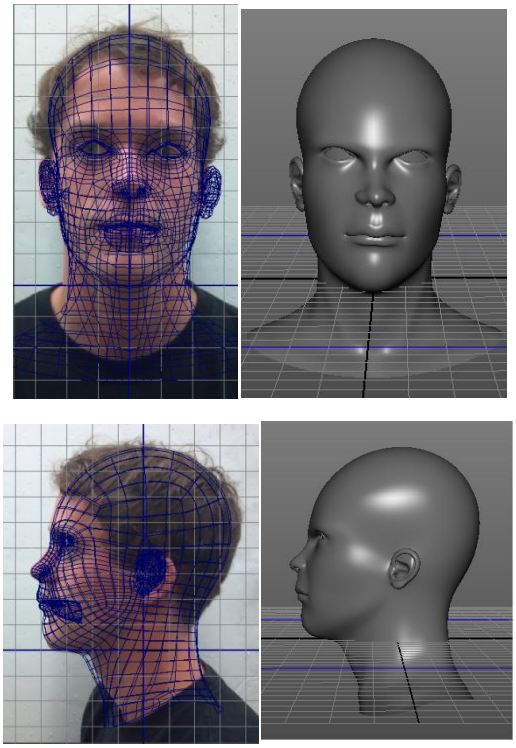

Figure 17: *The resulting mesh in Maya, shown in wireframe and shaded views. The shape of the model could be improved with additional landmark points.*

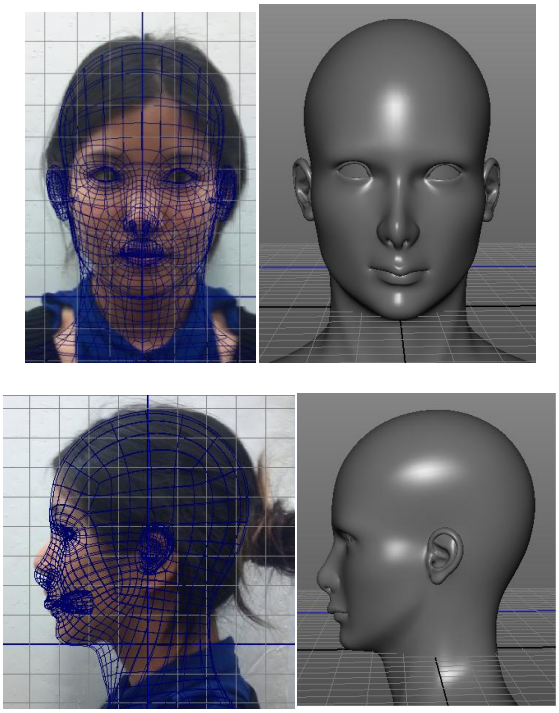

Figure 18: *A second example of a final model.*

References

[1] AKIMOTO, T., SUENAGA, Y., AND WALLACE, R.S. 1993. Automatic Creation of 3D Facial Models. In *IEEE Computer Graphics and Applications*, pp. 16-22.

[2] CANNY, J. 1986. A Computational Approach to Edge Detection. In *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8(6):679–698.

[3] NAKATA, Y. AND ANDO, M. 2004. Lipreading Method Using Color Extraction Method and Eigenspace Technique. In *Systems and Computers in Japan*, vol. 35, no. 3.

[4] SHI, J. AND TOMASI, C. 1994. Good Features to Track. In *CVPR*, pp. 593-600.

[5] TANG, L. AND HUANG, T. 1996. Automatic Construction of 3D Human Face Models Based on 2D Images. In *Proc. Int. Conf. Image Process.*, 1996, vol. 3, pp. 491–520.

[6] VIOLA, P. AND JONES, M. 2001. Rapid Object Detection Using a Boosted Cascade of Simple Features. In *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recog.*, pp. I-511–I-518.