2D Transformation

Based on *Fast Grid-Based Nonlinear Elasticity for 2D Deformations*

Mai Trinh
COSC 6372 Computer Graphics*
University of Houston

May 6, 2021

1 Introduction & Problem Statement

Image deformation is a popular technique widely used in animation, photography, special effects and medical imaging. Image deformation is a process of manipulating an image such that shapes portrayed in the image is distorted. A general idea is to determine the way in which the pixels in the original image should be mapped to the pixels in the target image. So, we need to specify how each pixels moves between the two locations. This could be done by specifying the mapping for a few important pixels. The motion of the other pixels could be obtained by appropriately extrapolating the information specified for the control pixels. These sets of control pixels can be specified as points, lines, curves, or grids to produce a smooth and realistic deformed image.

Image deformation is a popular area of research in computer graphics, computer vision, thus there has been much previous work on image deformation. Thus, there have been many research presenting various image deformation technique to create realistic image deformation. I briefly discuss those works that is closely related to this project. The pioneer work of [1] presented the concept of guiding a warp using feature lines. A method that is commonly used in image deformation is Moving Least Squares. [2] provided an image deformation technique based on cubic splines and moving least squares.

This project uses an approach based on *Fast Grid-Based Nonlinear Elasticity for 2D Deformations* [5] which presented a new method for image deformation using user-specified contour pairs to indicate the original and target shapes of important feature lines. The paper is organized as follows: I briefly summarize the key attributes and proposed method of the referenced paper in the first part. The second part describes in detail my implementation of the image deformation technique. The third part presents the experimental results of our approach. Lastly, conclusion and references are given in the last part of the project.

2 Paper overview

In this section, I will summarize the paper and their approach to the image deformation technique. In this paper, the author proposed a new method for image deformation by using spline curves to specify the starting and the target shape of the contour lines. The method leverages a custom nonlinear elastic membrane model to generate a 2D deformation map from these contour lines. With the use of elastic model, it provides the ability for local area preservation and the accommodation for irregular-shaped geometric constraints. Using a regular, standard Cartesian grid, they compute the deformation field with no restrictions on the constraint curves. They allow for user-imposed constraints to be resolved at sub-grid resolution without the need for re-meshing. The regular discretization facilitates the use of highly efficient multi-grid solver for computing the deformed image shape. Figure 1 displays an example of warping method proposed by this paper. This example shows an elastic sphere being pinched between 2 fingers. In the left image, the red line is identified as the starting contour line, and the blue line is the target shape of the feature line. The effect was created by providing various compression resistance value to make the sphere significantly more area preserving that its background. Since the sphere is being pinched, pixels in the middle part of the sphere is compressed, thus to compensate for compressed area, pixels of the surrounding area are stretched (this effect is displayed more clearly in the right image of figure 1. Elastic membrane model is able to stretch to create separation and compress when image contraction is desired and bulge to approximate area preservation.

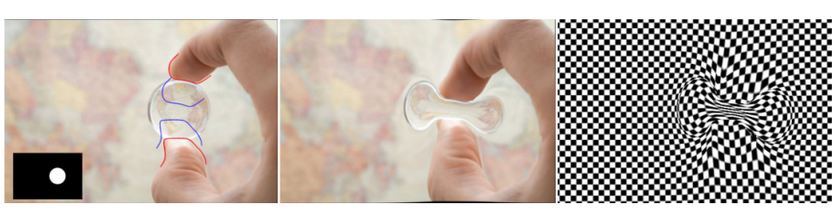

Figure 1: Image warp of an elastic sphere being pinched between two fingers.

*Final project report

Figure 2 shows an example of the in-compressible image warping in a natural squash-and-stretch effects. Figure 3 displays another example of the image deformation method presented in this paper. The ability to impose displacement constraints along arbitrary spline curves which do not have to be aligned with nodes on the regular Cartesian grid as shown in this example. The accurate closing of the mouth is enabled by precisely targeted curves designed to move the lip upward while keeping the teeth in place. The proposed approach is to encode compression resistance value directly into the model, thus imposing a soft constraint on the deformation map. This method enables graceful degradation when the user-specified deformation create folding artifacts and enables local area preservation by specifying spatially-varying compression resistance values for corresponding areas.

Figure 2: Squash-and-stretch deformation of a boxer face

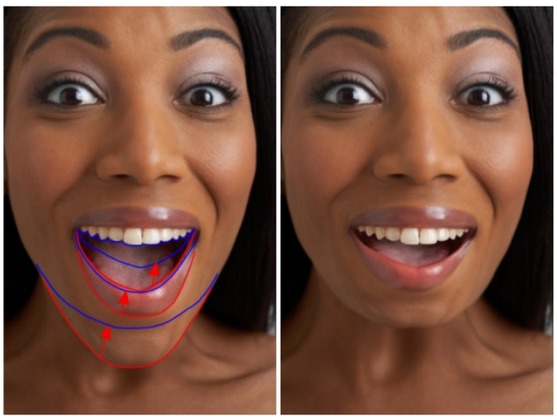

Figure 3: Image warp portraying the closing of the mouth.

Overall, this paper presented a new method for image deformation that combines speed and accuracy. High-quality and realistic image warping can be achieved using user-specified contour pairs on an elastic model and allow area preservation.

3 Implementation

This section describes my implementation of the image deformation technique based on the previous work. The implementation is done using Python 3.9, OpenCV, and numpy library. First, the application loads an image and determine the contour lines. A contour line is simply a curve joining all the continuous points long the boundary, having same color or intensity. In OpenCV, finding contours is similar to finding white object from black ground. Since finding contours of a binary image produces better accuracy, so I first convert the original image to a binary image by disabling (setting to 0) pixels that has a value less than 255 and any pixel that has a value greater than 255 is set to 255. Before finding contours, I apply edge detection using Canny algorithm. Edge detection is used to find the boundaries of objects within images. Usually, it works by detecting discontinuities in brightness. I use Gaussian blur algorithm to filter out noise. Then use the provided Canny algorithm from OpenCV, I identify the edges after fine tune the threshold values. Note that the threshold values are different for each image, most image works well within low threshold value of 85 and high threshold value of 255. The current implementation requires manual fine tuning for these threshold values. In the future, given more time I would implement a function that automatically identifies the optimal threshold values for the input image. Once the edge detection is completed, I can find the contours and draw all contours into the original image. Figure 4 displays the process of fine tuning threshold values for Canny edge detection algorithm using 2 track bars (one for low threshold value and one for high threshold value) and the result detected contour lines on the original image. The above implementation is performed using the tutorial described in [3] and [4].

The application then displays the image with the highlighted feature lines in an interactive window allowing the user to specify the starting and the target shape of the feature lines. Key points are set to create control curves; the curves are moved to new positions to guide image warping. For each identified curve, the user must choose at least 3 points along the feature lines where they want to use. The longer the curve, the more points are needed to identify the shape of the curve correctly. Then, the application forms a curve from connecting these points. Figure

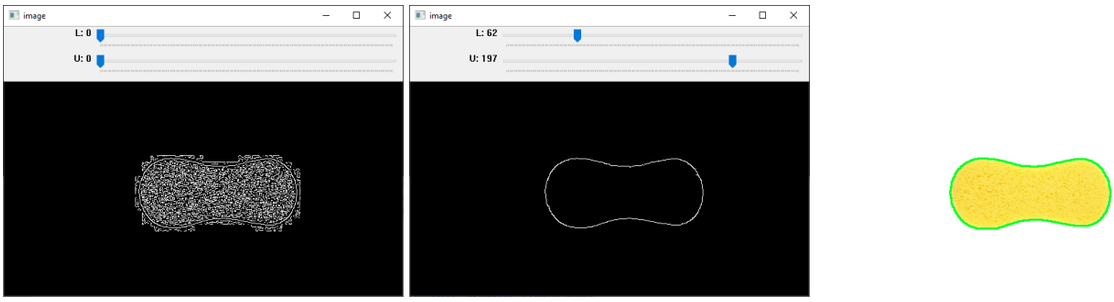

Figure 4: Fine tuning threshold value for Canny edge detection

5 shows another example of this implementation. The left image is the result of applying edge detection algorithm after performing fine tuning, and the right image displays the contour lines in green and the user-specified contour pairs. The contour pairs are identified as follow: the red line is the starting line and the blue line is the target shape of the contour line.

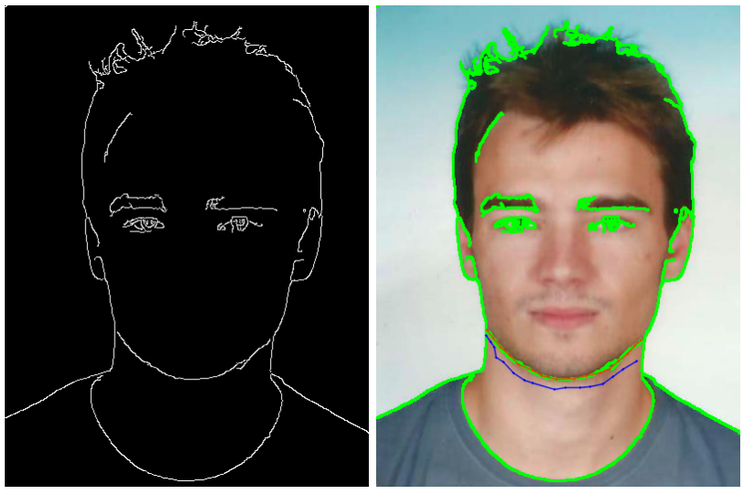

Figure 5: Edge detection, contour lines and user-specified feature lines

Finally, the application takes the value of user-input feature lines and apply the warping method on the image. The warping method utilizes the mapping function from the original positions to the destination positions. The mapping function calculates the shifting vector (i.e. the distance and the direction) from original point to the destination point. The neighboring pixels move less than the control point and in various amount. The amount of movement for the neighboring pixels is specified by a weighting function depended on the length of each shift vector and the distance from the pixel to the shift vector. As the pixel is further away from the control point, the amount of movement reduces. Then, it uses a resampling function to compute the new coordinates of every pixel and copies the color of the source pixel. Note that the pixels that are far away from these contour lines remain unchanged since the weight for these pixels are 0. Thus, we don't actually compute new coordinate for these pixels because the coordinate of these pixels are being applied by the weight value of 0, which means it does not change.

4 Deformation Results

Based on the implementation above, I tested the image deformation on a number of images and compared the resulting images.

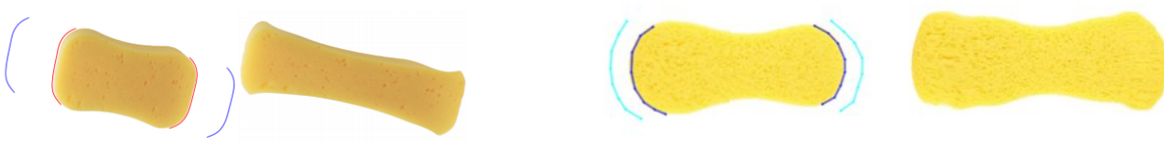

Figure 6: Stretchy sponge used for performance tests.
Example from the paper (left), resulting deformation (right)

Figure 6 shows an example of image deformation on a sponge. The image on the left is the warp example taken from the referenced paper [5], and the right image is the resulting deformation image from my implementation. Note that the input images of the 2 examples are different since I was unable to find the exact image used in the paper. Thus, the comparison is only relatively to understand the deformation behaviors. The deformation of the stretchy sponge shows a relatively good result compare to the original image. Figure 7 shows another example of

the image deformation to deform the man's face. The method can express the character's feature lines and make his face deformation smooth and realistic. Note that in this example, the edge detection algorithm also detect the hair line, eyes, and eyebrows as the contour lines. Thus, further optimization of edge detection algorithm is needed to improve performance.

Figure 7: Image deformation based on control curves.

5 Limitation

While the deformation technique shows reasonably good result for simple deformation, the implementation produces some artifacts for more complex deformation. Figure 8 shows an example of such artifacts. In this example, I attempt to move the thump closer to the rest of the hand, however the result shows the image is bend. I suspect the reason for such distortion is that when the control points are close together but move in different directions or amounts, the image would locally swirl and bend around them. To fix this problem, the spatially-varying compression resistance value should be taken into account with an embedded sub-grid spline constraints allow for accurate targeting. Due to the time constraint to complete the project, these features have not been added into the current implementation.

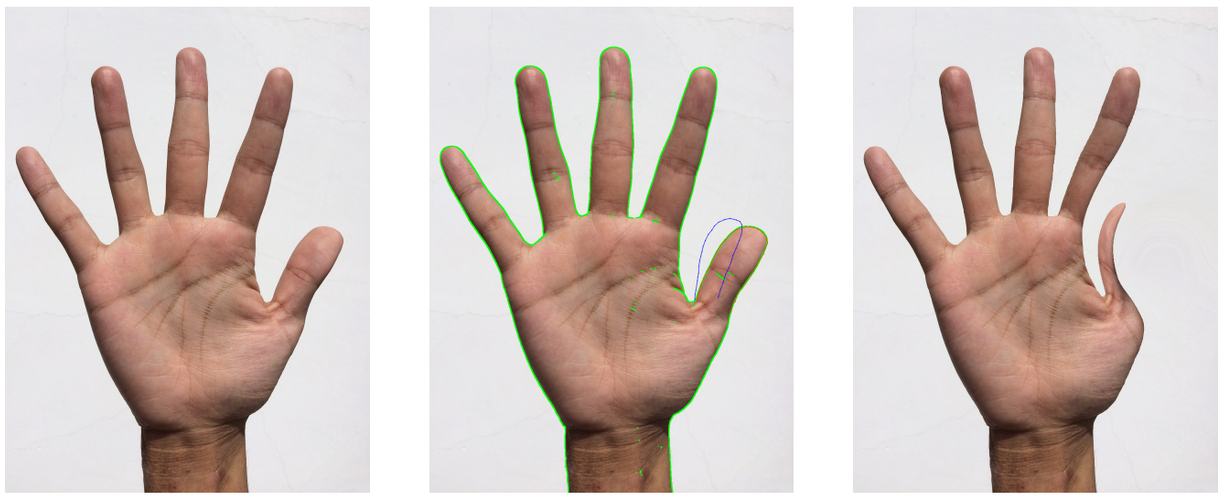

Figure 8: Image deformation on complex distortion resulted in image bending

6 Conclusions & Future work

In this paper, I presented the implementation of the image deformation technique using contour lines for expressing the original and the target shape to control the deformation. Generally speaking, the image deformation technique using contour curves produces reasonable results for simple deformations such as stretch and compress. In term of limitations, this method suffers from fold-backs and bending artifacts like most other space warping approaches. These situations occur when performing more complex distortions. The reason for such distortion is the lack of local area preservation and constraints. Thus, in future work, I would like to implement spatially-varying compression resistance value and impose displacement constraints to enable local area preservation. Moreover, this project is not suitable for heavy image processing as it is not optimized to process large amount of data. Additional implementation to allow parallelization or the use of multiple CPU cores can significantly improve the computation performance.

References

[1] Thaddeus Beier and Shawn Neely. "Feature-Based Image Metamorphosis". In: *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '92. New York, NY, USA: Association for Computing Machinery, 1992, pp. 35–42. ISBN: 0897914791. DOI: 10.1145/133994.134003. URL: https://doi.org/10.1145/133994.134003.

[2] Shungang Hua, Shaoshuai Li, and Xiaoxiao Li. "Image Deformation Based on Cubic Splines and Moving Least Squares". In: *2009 International Conference on Computational Intelligence and Software Engineering*. 2009, pp. 1–4. DOI: 10.1109/CISE.2009.5366816.

[3] OpenCV. *OpenCV Contours: Getting Started*. URL: https://docs.opencv.org/master/d4/d73/tutorial_py_contours_begin.html.

[4] Abdou Rockikz. *How to Perform Edge Detection in Python using OpenCV*. URL: https://www.thepythoncode.com/article/canny-edge-detection-opencv-python.

[5] Rajsekhar Setaluri et al. "Fast Grid-Based Nonlinear Elasticity for 2D Deformations". In: *Symposium on Computer Animation*. 2014, pp. 67–76.