

# Doe, John G.

## Automatic Image Correspondence for Video Clips

Faculty Mentor: Joseph Cougar, Electrical and Computer Engineering

Image correspondence is a powerful method for matching features in two images. It has a wide variety of applications, and can be used to automatically perform complex image manipulation tasks that have previously been done by hand. In this research, image correspondence has been used to manipulate video clips, including morphing and slow motion techniques.

The basic problem of image correspondence is this: given two images, find the areas of each image that match with the other. For example, given two faces, a good image correspondence algorithm can match up the eyes, noses, mouths, ears, and other facial features. If it can't find an exact match, it should find the "best" match.

Image correspondence is a difficult problem, because it is difficult to define mathematically. One must decide exactly how to measure the quality of a correspondence, and many measures have been used.<sup>1</sup> For this reason, it is unproductive to define the problem in terms of a correct correspondence; rather, we are searching for the best possible correspondence using a given metric.

Our technique uses a triangular mesh for finding image correspondence. We do this by creating two identical meshes, one on each image (Fig. 1). We compare the contents of each pair of triangles, one in each mesh, to see if the internal colors and shapes are similar. By adding new triangles to the mesh, and by moving their corners to better locations, we can gradually reduce the cost according to our metric, and find a correspondence.

One application of this technique is making video morphs. Video morphs work by taking two video clips of equal length, and then determining correspondences between each of their frames. To make the morph, you simultaneously advance the time constant for interpolating the meshes and the frame number for playing back the clip. The end result is that the first video clip seems to turn into the second.

This technique does create a morphing effect on moving images, however it has a difficult problem. The algorithm searches for corresponding areas in each frame pair, and since these frames are different, the solutions to the correspondence problem can vary slightly. Our initial algorithm treated each frame as a separate case, and decided on which areas match up without consideration of how those decisions were made for previous frames. This results in discontinuities between frames, with parts of the images appearing to jump around erratically. This effect is termed jitter.<sup>2</sup> We developed a technique for measuring the amount of jitter between frames, and added the jitter measurements to the morphing algorithm, penalizing any jitter that occurs. This works well for some clips, but does poorly if there is a lot of motion in the clips themselves.

A better solution to jitter is to resolve the correspondences between frames and clips at the same time. However, this involves the ability to have many images loaded into the program at once, and the ability to resolve morphs between many images. We are currently working on adding these capabilities.

Another application for automatic image correspondence is slow motion. One may create a slow motion clip by finding correspondences between each frame and the one following it. By rendering the morphs in order, you effectively create a slow motion version of the video clip. This technique has a nice advantage, in that you can slow down the motion as much as you want—simply increase the frame count when you make the morphs.

Initial results with the slow motion technique were very encouraging. Our algorithm accurately captures the motion of objects in a video clip, and on clips where the correspondence is of good quality; the resulting slow motion clips look good (Fig. 2). We have found a few situations for which the correspondence is poor. One is translation at image boundaries. Objects that slide rapidly off the frame blur and look bad, since the algorithm can find no matches between the frames for them. The second problem, opposing motion, is when two overlapping objects move in different directions. Our mesh does not handle this kind of motion, since we can't "rip" the mesh.

The problem of edge translation was effectively solved by placing a blank, "neutral cost" border around the images. The cost of this area is determined by averaging the cost of surrounding triangles. Thus, whenever the algorithm tries to move parts of the mesh into the neutral cost zone, it neither helps nor hurts the overall cost. This effectively allows objects to slide off of the edge easily, without encouraging it when it is unnecessary.

In the future, we wish to solve the problem of jitter in video morphs, and allow for opposing motion in slow motion clips. Our software is capable of handling these problems if the data is input by hand. By analyzing the nature of these problems, we can find algorithmic solutions in the future.

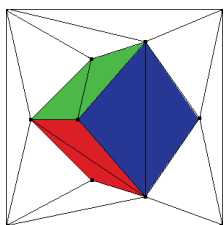


Figure 1 – An example triangular mesh (the black lines). This mesh was created in our lab using software written by Dailin Ge. Note the triangles outlining the interior shapes, and note how triangles connect at the corners.

Figure 2 – A sample from a slow motion clip. The center image was synthetically generated. The skier is moving off the edge of the frame, so this image used a neutral cost border.



<sup>1</sup> For example, in our research we have used RGB color difference, bending cost, triangle area, average color, gradients, and planar fits as methods for comparing images. Many other techniques are possible.

<sup>2</sup> Our initial attempt at video morphing clearly demonstrates the problem of jitter. A copy of the resulting video clip is online at the time of writing. The URL is <http://students.cs.byu.edu/~ahelps/jitter.mov>.