

ADVANTAGES OF THE CHOSEN ALGORITHMS AND FUTURE IMPROVEMENTS*

Jennifer Gillenwater
J. Ryan Stinnett
Elica Skorcheva

This work is produced by The Connexions Project and licensed under the
Creative Commons Attribution License †

Abstract

This module addresses several possibilities for the extension and optimization of basic image registration and interpolation algorithms.

As far as registration is concerned, the method we chose was the most inclusive one. It enables us to have LR frames that are a zoomed, translated, panned, tilted or rotated versions of each other. However, since it is based on derivatives at each pixel, it usually gives poor results with images that are dramatically different from each other. To get around this issue, we could use additional techniques that detect large-scale differences before running the current registration algorithm, which would then refine that data.

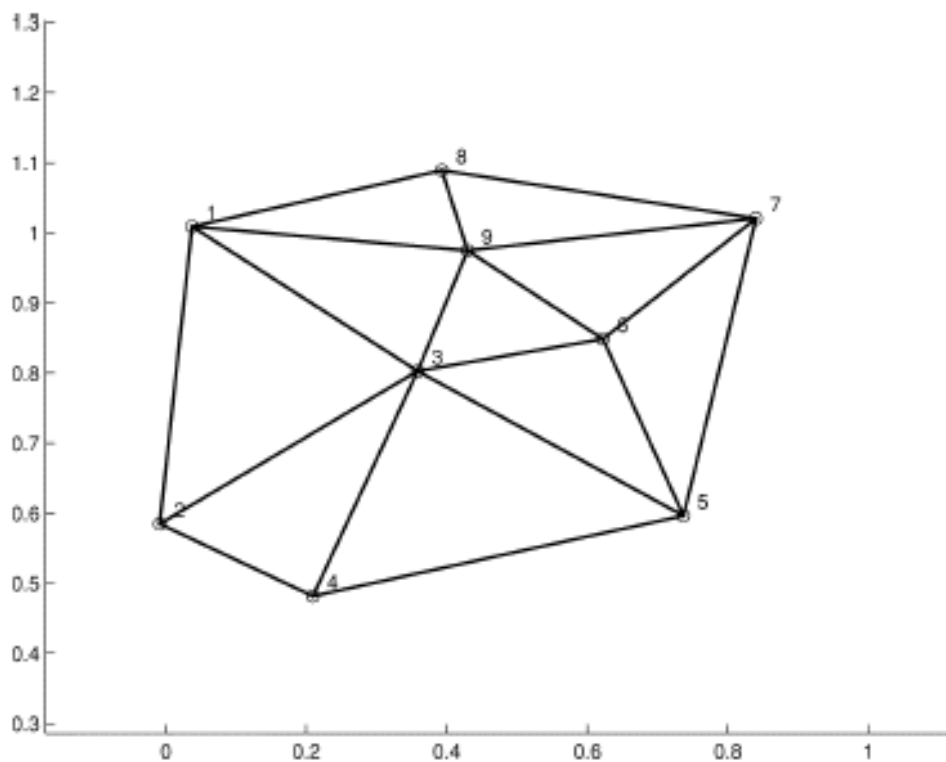
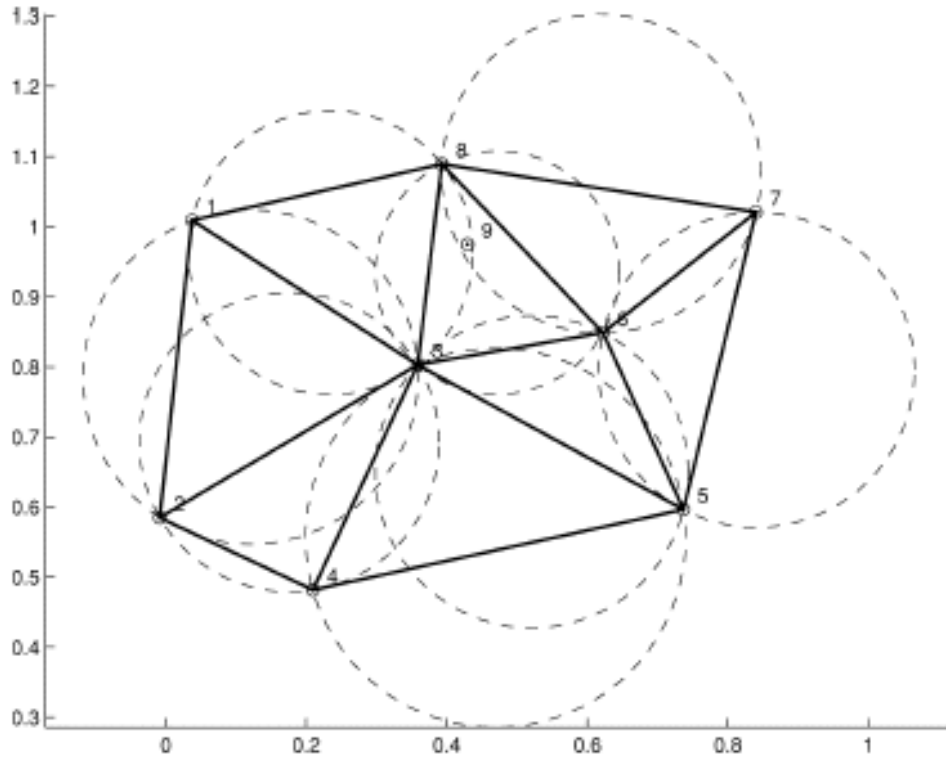
The interpolation algorithm that we implemented, though it takes advantage of Delaunay triangulation, uses averaging to determine new pixel values. A more accurate interpolation could be obtained by approximating each triangle patch by a continuous function, such as the bivariate polynomial described in the Bose-Lertrattanapanich Pixel Interpolation module. This would at the end generate a better HR image.

Nevertheless, our implementation still derives several of the same benefits as the Bose-Lertrattanapanich from the Delaunay triangulation model. In particular, one of these benefits is the option of implementing an efficient update algorithm for an HR image. The ability to update is important in that it allows more data to be incorporated into the final image should additional LR images become available after the original HR image was constructed. In this respect, Delaunay triangulation is very efficient because the triangulation can be updated on a local basis; new points can be inserted without destroying the entire original triangulation.

*Version 1.2: Dec 21, 2006 10:04 pm US/Central

†<http://creativecommons.org/licenses/by/2.0/>

Site-insertion Example



<http://cnx.org/content/m14215/1.2/>

Figure 1: Original triangulation (top), Updated triangulation with vertex 9 inserted (bottom). (Source: 2)

The algorithm for site insertion is described below:

1. Find which triangles' circumcircles enclose the first new vertex (these triangles are now no longer Delaunay).
2. Define an insertion polygon by finding the convex hull of the vertices that make up all of the triangles from (1).
3. Eliminate all edges from the original triangulation that lie inside the insertion polygon.
4. Connect each vertex of the insertion polygon to the inserted vertex with a new edge to generate a new triangulation.
5. Update the old triangulation to the new and loop back to (1) to insert the next new vertex.