An easy-to-implement numerical, visual method for generating the (non-)linear separating isosurface of two classes of objects in two-dimensional space using Marching Squares and OpenCV

Shawn Halayka*

Wednesday 27th April, 2022 22:14

Abstract

With regard to the separating isosurface of two classes of objects in two-dimensional space, the transition from curvilinear to rectilinear is documented. Marching Squares and OpenCV were used to calculate the data that were used for illustrations in this paper.

1 Introduction

In terms of statistical learning, a balance between high variance (curvilinearity) and high bias (rectilinearity) is to be had [1].

Marching Squares works as a radial (spherical) isosurface generator, as well as a rectilinear (planar) isosurface generator – it all depends on the grid resolution. The data generated by Marching Squares are the line segments that separate the two classes, as well as triangles for the area of each class. Classifying a test point is a matter of ray-triangle intersection, which can be accelerated using a Bounding Volume Hierarchy or quadtree data structure.

OpenCV was used to downsize images. OpenCV was also used to blur images.

2 Conclusion

Test...

*Independent – sjhalayka@gmail.com

References

[1] James, et al. An Introduction to Statistical Learning with Applications in R. ISBN: 978-1-0716-1417-4

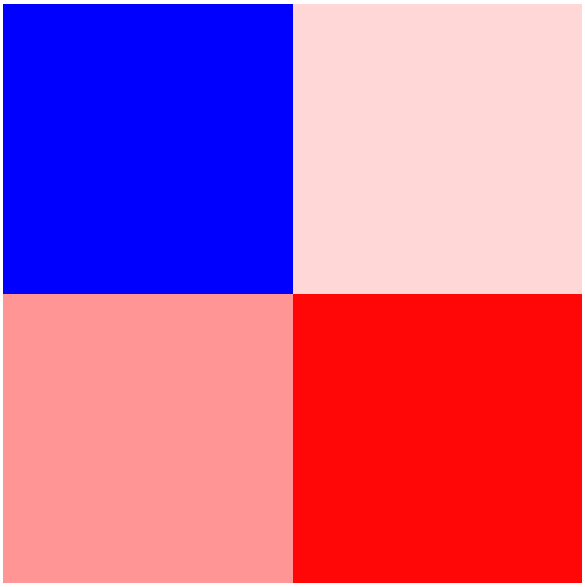

Figure 1: Bitmap image used as input to the Marching Squares algorithm. Image size is 2x2.

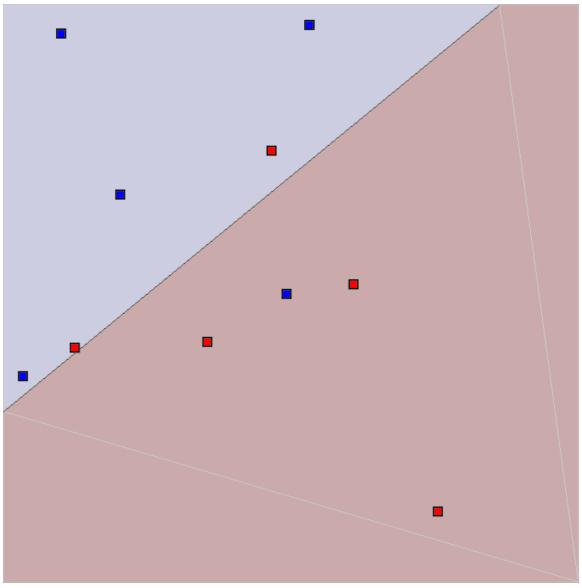

Figure 2: The output of the Marching Squares algorithm: rectilinear separation. Grid resolution is 2x2.

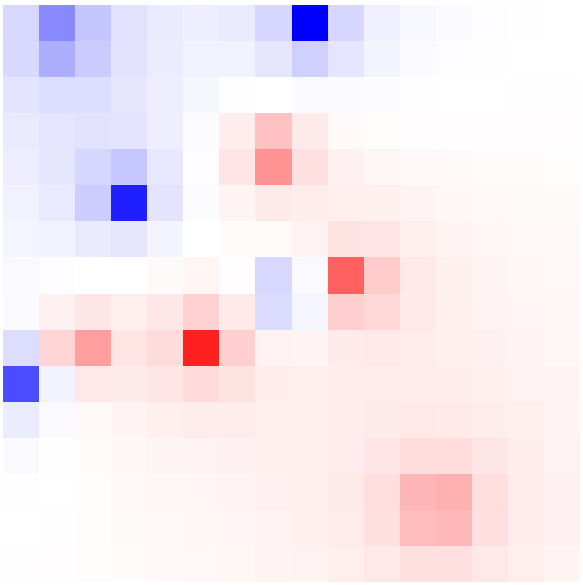

Figure 3: Bitmap image used as input to the Marching Squares algorithm. Image size is 16x16. Colour falls off with distance.

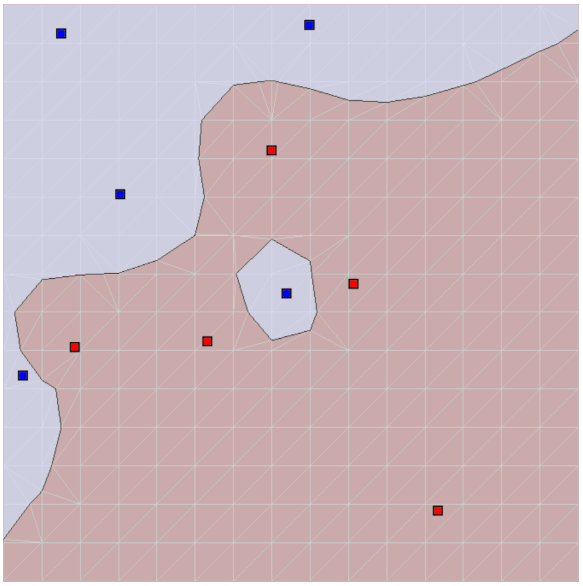

Figure 4: The output of the Marching Squares algorithm: curvilinear, radial separation. Grid resolution is 16x16.

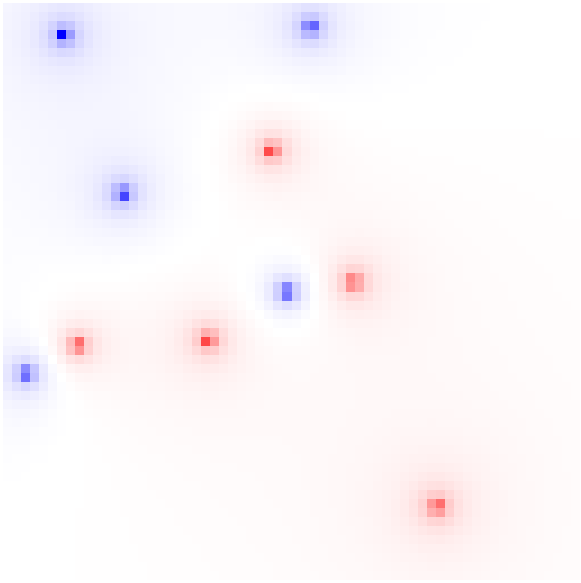

Figure 5: Bitmap image used as input to the Marching Squares algorithm. Image size is 64x64.

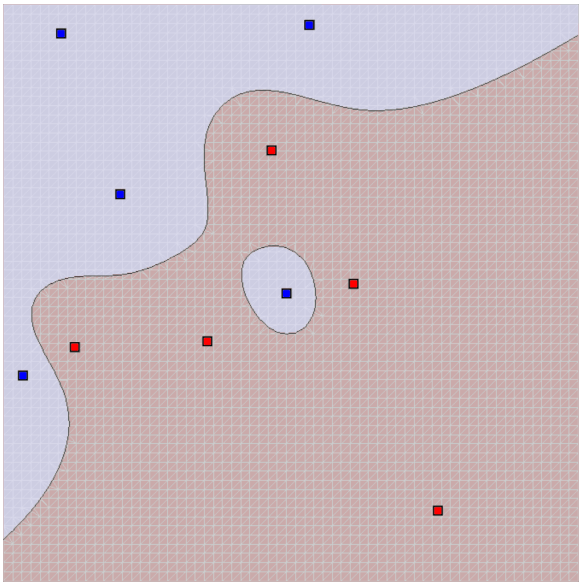

Figure 6: The output of the Marching Squares algorithm: curvilinear, radial separation. Grid resolution is 64x64.

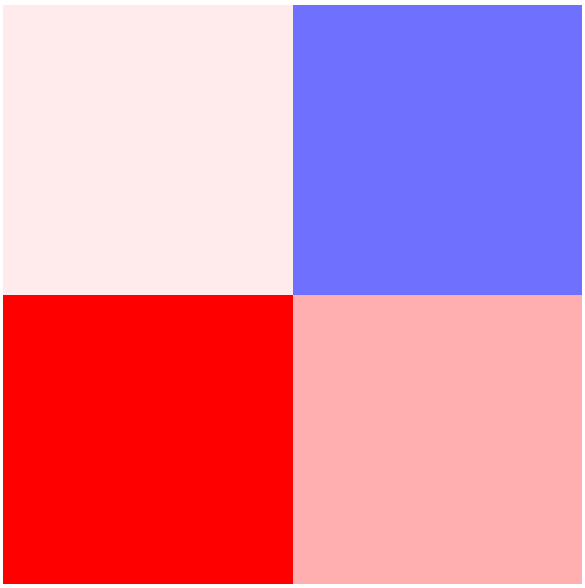

Figure 7: Bitmap image used as input to the Marching Squares algorithm. Image size is 2x2.

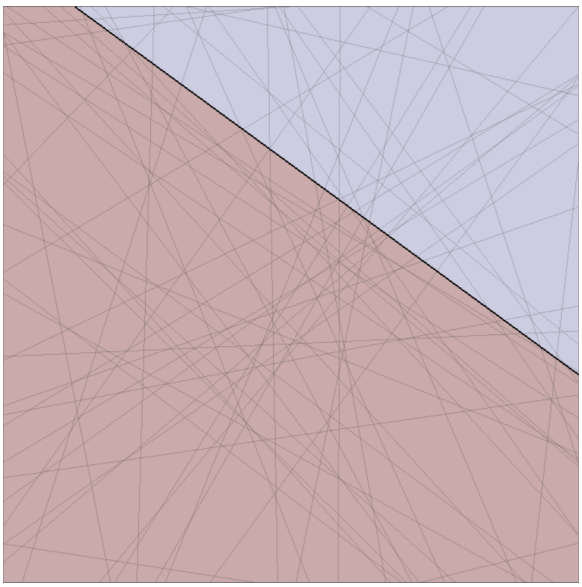

Figure 8: The output of the Marching Squares algorithm: rectilinear separation. Grid resolution is 2x2. All 50 contour sets have been drawn in transparent grey, with the exception of the average contour set, which is drawn in pure black.

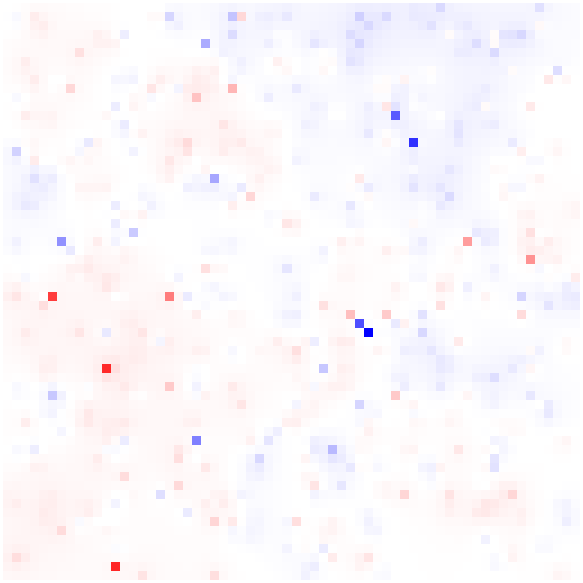

Figure 9: Bitmap image used as input to the Marching Squares algorithm. Image size is 64x64.

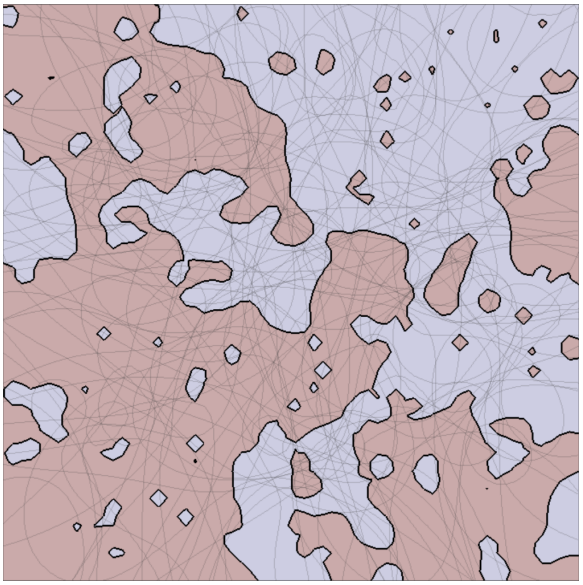

Figure 10: The output of the Marching Squares algorithm: curvilinear, radial separation. Grid resolution is 64x64. All 50 contour sets have been drawn in transparent grey, with the exception of the average contour set, which is drawn in pure black.

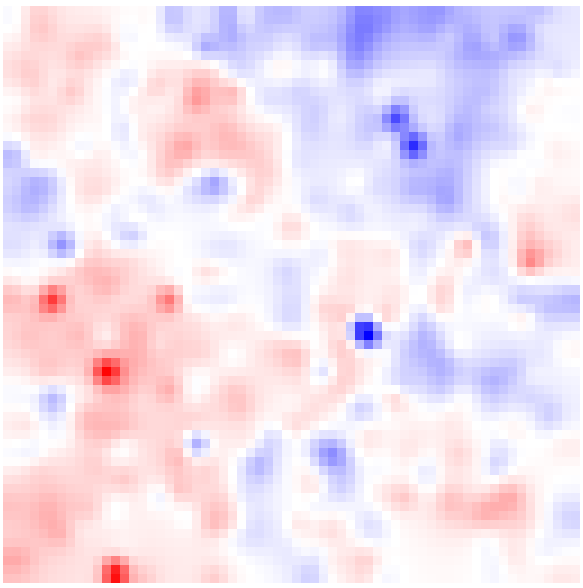

Figure 11: Bitmap image used as input to the Marching Squares algorithm. Image size is 64x64. One iteration of Gaussian blur has been used.

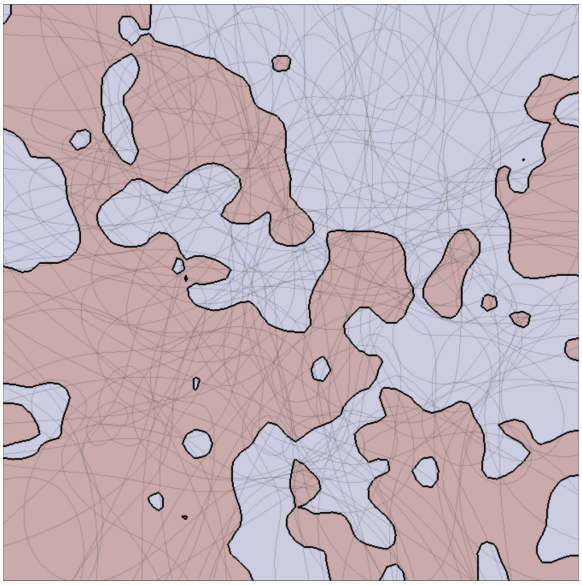

Figure 12: The output of the Marching Squares algorithm: curvilinear, radial separation. Grid resolution is 64x64. All 50 contour sets have been drawn in transparent grey, with the exception of the average contour set, which is drawn in pure black.

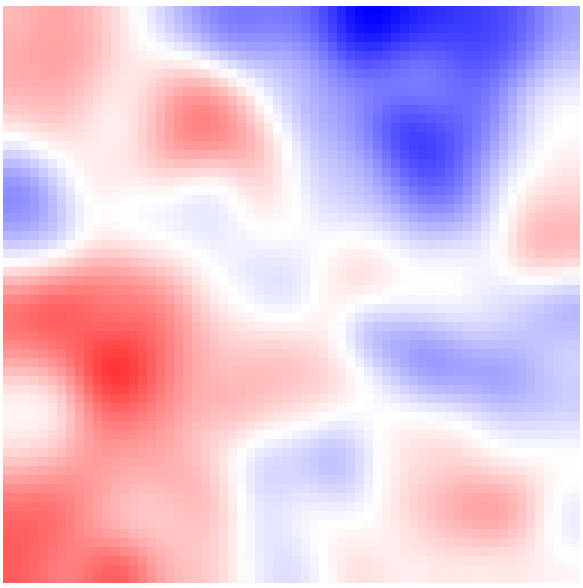

Figure 13: Bitmap image used as input to the Marching Squares algorithm. Image size is 64x64. Ten iterations of Gaussian blur have been used.

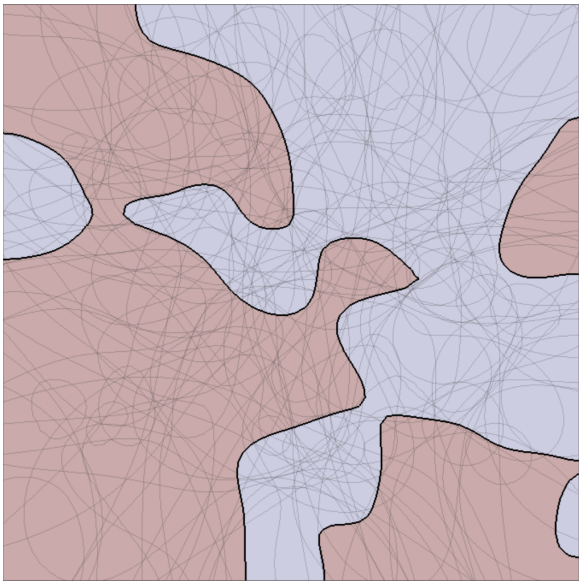

Figure 14: The output of the Marching Squares algorithm: curvilinear, radial separation. Grid resolution is 64x64. All 50 contour sets have been drawn in transparent grey, with the exception of the average contour set, which is drawn in pure black.