Connecting points by Smooth Connection Functions defines shapes in the smoothest possible way

Alex Alon[1] and Sven Bergmann[2]

[1] Chief Scientist at R.A.W
Sheshet Hayamim 11/c, Binyamina, Israel
[2] Associated Professor at the University of Lausanne
Department of Medical Genetics
Rue du Bugnon 27 (DGM-101), 1005 Lausanne, Switzerland

Abstract. We consider the basic challenge of drawing a smooth curve through a list of ordered points in a plane. This is similar to the kids' game known as "join-the-dots", which usually results in some kind of shape. Requiring that not only the curve but also its slope is continuous at each point gives rise to a well-defined mathematical optimization problem. In order to address it we make use of co-called "Smooth Connection Function" (SCF), which provide generic solutions to Hermite's interpolation between two points in two dimensions. We have shown previously that SCFs can be determined analytically as the exact solutions of a variational problem that is invariant under translations and rotations (J. Phys. A: Math. Gen. 35 (2002) 3877). A SCF has a unique solution, given its boundary points and slopes, and only depends on a single parameter ν determining the relative importance of minimal curvature with respect to minimal length. The extreme examples are a very large ν, requiring minimal curvature at each point and therefore giving rise to curves made only of circle segments, and a vanishing ν optimizing only total length and thus resulting in straight lines. Here we show that the curve connecting all points can be constructed from SCF segments and that the slope at each point arises naturally when optimizing overall smoothness. Our approach provides a practical, yet rigorous, solution to the very general challenge of defining shapes from a collection of points.

1 Introduction

Shapes can be defined in many different ways. While for arbitrary shapes, it may be impossible to define them with a finite list of elementary curves or drawing instructions [1], it is often possible to approximate shapes with such a list. Such an approximation may suffice for practical purposes because of the limited resolution of human vision. For example, digitalized images often provide reasonable representations of real shapes (which may be as complicated as those corresponding to human faces). Yet, pixelized images do not scale, in the sense that there is a fixed length scale beyond which zooming into the image will provide no further details (although they might have been present in the original object).

Representation of shapes by drawing instructions is one possibility to overcome this limitation. Usually such instructions include a list of points together with a prescription of how to draw curves between some of these points. This is also known as vector presentation and a well known example are letters in true type fonts [2]. The simplest way to connect between points is to draw straight lines, but this will necessarily give rise to discontinuities in the slope of the connection curve at the points (see Fig. 1). Using higher order polynomial functions provides a simple way to overcome this limitation. For N points in two spatial dimensions one can always find a polynomial of degree $N-1$ that goes through all points. Yet, spline interpolation is usually preferred to this polynomial interpolation because it allows for determining in which order the points are passed by the curve, while avoiding oscillations of the curve (known as Runge's phenomenon [3–5]). A spline is a sufficiently smooth piecewise-polynomial function [6]. For each piece it is usually sufficient to use low-degree polynomial functions. For example, the four coefficients of cubic polynomials are fully determined by the coordinates of two points (with different x-coordinates) and requiring that the slopes and curvatures at these points match to the neighboring pieces. Yet, the resulting curves depend on the choice of coordinate system and their shape is not invariant under rotational transformations. Parametric Bezier-curves [7–9] provide an answer to this by explicitly conserving rotational invariance. This is because they are defined through the two end points as well as a number of control points, such that a rotation of these points will also rotate the entire shape defined by the curve they generate. The most common cubic Bezier curves have two control points and are thus defined by eight parameters.

This allows for adjusting not only the slope of the interpolating curves at the end points, but also its curvature. However, there is no principled way for choosing the control points other than minimizing the deviation from some desired curve. In a work we published almost ten years ago [10] we proposed a new class of functions that interpolate smoothly between two given points [11–15]. These so-called "Smooth Connection Function" (SCF) are the exact solutions of a variational problem [16, 17] that is invariant under translations and rotations. Specifically SCFs minimize the curvature κ raised to some power ν along the

curve C in the sense that the integral (corresponding to a cost-function)

$$S = \int_C [\kappa]^\nu \, ds \,. \tag{1}$$

takes the smallest possible value. Thus the choice of the parameter ν determines the relative importance of minimal curvature with respect to minimal length. Each choice of ν defines a class of SCFs. The extreme cases include a very large ν, requiring minimal curvature at each point and therefore giving rise to curves made only of circle segments, and a vanishing ν optimizing only total length and thus resulting in straight lines. Intermediate values of ν give rise to compromises between these extremes.

Here we set out to use the optimization criteria in eq. (1) to establish a generic approach for connecting an ordered set of points with a single curve. Our procedure allows for defining shapes only based on points, while the slopes and the curvatures at these points are determined automatically in such a way that the "global cost" S_Σ is minimal for a given ν. $S_\Sigma = \sum S_i$ is defined by the sum of S_i corresponding to each curve segment C_i. Importantly, while each segment is a SCF, the slopes at the points are obtained by an iterative optimization procedure. The choice of ν in general impacts the resulting shape. For example, in the case of three points, $\nu = 0$ gives rise to the triangle having these points at its corners, while an infinite ν results in the circle having these points on its circumference. In other words, our method provides a solution to the game known as "join-the-dots", by drawing a continuous, and for $\nu < 0$ or $\nu > 1$ smooth, curve along the dots (see Fig. 1).

Before we describe our optimization procedure and show a few examples of how the choice of the single parameter ν alters the shape which is otherwise defined by the same collection of points, we would like to provide some further motivation for the usefulness of our approach. In our previous work [10] we showed that SCFs can define optimal trajectories also in an economic sense. For example, we considered a (space) ship that starts from some point with a given orientation and has to reach a target at a given angle. We argued that its total fuel consumption is proportional to the cost function in (1) if directional changes are made by exerting a force F perpendicular to its motion with a fuel consumption per unit time W that is governed by some potential law ($W \propto F^\nu$). In the case of multiple points the corresponding scenario could be a list of targets that need to be passed in a given order. Our approach would yield the optimal trajectory of approach for each target.

2 Results

We start by introducing some notations: We call the function $y(x)$ that interpolates between an initial point $\boldsymbol{P}_i$ and a final point $\boldsymbol{P}_f$ with (Euclidean) coordinates

$$\boldsymbol{P}_i = (x_i, y_i) \qquad \text{and} \qquad \boldsymbol{P}_f = (x_f, y_f) \,, \tag{2}$$

a "Smooth Connection Function" (SCF), if it minimizes S in eq. (1) while satisfying the boundary conditions

$$y(x_i) = y_i \qquad \text{and} \qquad y(x_f) = y_f\,, \tag{3}$$
$$y'(x_i) = \tan(\alpha_i + \alpha_0) \qquad \text{and} \qquad y'(x_f) = \tan(\alpha_f + \alpha_0)\,. \tag{4}$$

Here the prime denotes a derivative with respect to x and α_0 is the angle between the positive x-axis and the vector pointing from $\boldsymbol{P}_i$ to $\boldsymbol{P}_f$. Since S only depends on the local curvature

$$\kappa[y(x)] = \frac{1}{r[y(x)]} = \frac{|y''(x)|}{[1 + y'(x)^2]^{3/2}}\,, \tag{5}$$

and the infinitesimal length element along the curve

$$ds(x) = \sqrt{1 + y'(x)^2}\, dx\,, \tag{6}$$

it is invariant under translations and rotations.

For the functional parametrization of the interpolating curve the integral introduced in eq. (1) reads:

$$\tilde{\mathcal{S}}[y(x)] \equiv \int_{x_i}^{x_f} (\kappa[y(x)])^\nu \; ds(x) = \int_{x_i}^{x_f} \frac{|y''(x)|^\nu}{[1 + y'(x)^2]^{\frac{3\nu-1}{2}}}\, dx\,. \tag{7}$$

In [10] we showed how to find explicit solutions for $y(x)$ for a given ν subject to the boundary conditions in eq. (3) and eq. (4) making use of variational calculus [18] and the Noether theorem [19]. Several examples for SCFs given in Fig. 2. We also showed that there is a simple way to compute the cost function in eq. (1) for a given solution.

For the problem we consider in this paper the slope α_k assigned to each point P_k affects two curve segments. For the incoming curve $\alpha_k = \alpha_f$ and for the outgoing curve $\alpha_k = \alpha_i$. Thus the total costs S_Σ corresponding to the entire curve is a function depending on all α_k, and in general there is an optimal combination of these slopes that minimizes S_Σ.

We first implemented the following optimization procedure for computing the optimal curve (in the sense described above) for an ordered list of points in two dimensions and a fixed parameter ν:

1. For each point k we compute an initial guess for the slope in terms of the inclination of the line connecting the neighboring points: $\alpha_k^{(0)} = atan2(y_{k-1} - y_{k+1}, x_{k-1} - x_{k+1})$. (For closed curves this is always possible, otherwise for the initial and final points we use the lines connecting these points to their neighbors.)
2. For each point we compute the combined cost $S_k^{(n)}$ for the two curve segments that contain this point. (These are the only segments affected by the slope at this point.)

3. We alter slightly the slope in both directions at each point by $\pm\Delta\alpha$ and compute again the corresponding combined cost $S_k^{\pm}$ for the two neighboring curve segments.
4. We only update the slope at the point $\hat{k}$ yielding the *maximal* decrease in cost: $\alpha_{\hat{k}}^{(n+1)} = \alpha_{\hat{k}}^{(n)} \pm \Delta\alpha$.
5. We continue steps (2)-(4) until no change at any of the points leads to a decrease of S_Σ.

This is a greedy procedure, which is guaranteed to converge to a local minimum. Indeed, for the cases with relatively few dots we studied (see Fig. 3), we observed very nice convergence. The curves in Fig. 3 show how the interpolation curve evolves under the iterations. The initial curve is shown in black and the final, optimal curve in red. Intermediate curves are thinner and shown with colors ranging from blue to red. We investigated three lists of points: The top panel corresponds two three points, the middle panel to four points, and the bottom panel of 10 points. For the triplets we found that indeed for $\nu = 20$ our procedure finds an interpolating curve very close to a circle, as is expected for very large ν. For smaller values the total length of the curve becomes smaller, but this is at the expense of a larger curvature close to the three points. Interestingly, for small ν there is little adjustment of the slopes under the iterations. A similar behavior is observed when connecting four points. For 10 points, only some angles undergo significant changes during the iterations, while others stayed close to the initial guesses determined in step (1) above.

Trying out the greedy optimization procedure for the Octopus example in Fig. 1, we observed a rather long runtime, while the interpolation curve did not change dramatically for most points. We thus suspect that our simple optimization procedure gets stuck in a local minimum when applied to a large number of points. We therefore tried out a more sophisticated Black-box optimization tool using Gaussian Adaptation [20, 21]. We found that this algorithm seemed to work much better in the case of many points, making it applicable also for large numbers of dots. The evolution of the interpolation curve for the Octopus characterized by 95 dots is shown in Fig. 4 for $\nu = 2$ and $\nu = 20$. One can see that for the latter choice the interpolation curves tends to be rounder, in particular at the end of the tentacles. Yet, for close dots there is usually not much flexibility for adjusting the local slope. We therefore also tried out an example with only 58 dots, which resulted in much more refinements (Fig. 5). In particular for $\nu = 2$ the algorithm found that a looped curve at one tentacle allowed for reducing the action.

3 Discussion

Finding interpolating curves through a sequence of points is a fundamental mathematical challenge. While B-splines provide a solutions for generating smooth interpolations, they offer little control for the degree of smoothness. Our Smooth Connection Functions (SCFs) provide this feature in terms of a single tunable

parameter ν, but so far they could only be used for Hermite Interpolation [11], where also the slopes at the boundary of each curve segment have to be determined. Here we presented a general approach overcoming this limitation by automatically computing the *optimal* slope at each point. This optimization is based on a global cost function, summing up the contributions from each curve segment. For simplicity we assumed a single ν determining the relative importance between small curvature and a short path on all segments. Also we kept the slope at all points as free parameters subject to our optimization procedure.

Clearly, one could extend the present procedure by defining subsets of curve segments with distinct ν. Similarly, it is straight forward to fix the slope at some of the point. For example, this would make it much easier to provide an accurate presentation of the shape in Fig. 1 by fixing the angles at the end of the tentacles.

There are many conceivable applications for using our method for curve interpolation. For example, interpolation curves are a central tool in the field of Computer Aided Geometrical Design (CAGD) [9]. Yet, for computational reasons software for interpolation usually relies on simple functions, like the Bezier curves [7]. Tuning of these curves is cumbersome and requires many control points. In contrast, our procedure gives a principled approach for finding the best interpolation, which nevertheless is tunable. Finally, we would like to remark that one could in fact also perceive optimizing ν for a given shape and a fixed number of control points by minimizing the deviation of the interpolation curve from the real shape between these points. This would allow for assigning a typical ν to an entire shape or a part of it. For example, one could analyze handwritten letters and attribute a characteristic profiles of ν providing a unique signature to each handwriting. The feasibility of this and other applications will depend on further development and optimization of the computer code for the implementation of our method, which is available freely from the authors.

References

1. I. Johansson *Shape is a Non-Quantifiable Physical Dimension*, this workshop.
2. The US patents for TrueType fonts are available at: `http://freetype.sourceforge.net/patents.html`.
3. C. Runge *Über empirische Funktionen und die Interpolation zwischen quidistanten Ordinaten*, Zeitschrift fr Mathematik und Physik **46**: 224243 (1901), available at `http://www.archive.org/details/zeitschriftfrma12runggoog`.
4. J.P. Berrut and L.N. Trefethen *Barycentric Lagrange interpolation*, SIAM Review 46: 501517 (2004), doi:10.1137/S0036144502417715, ISSN 1095-7200.
5. G Dahlquist and A. Bjrk *4.3.4. Equidistant Interpolation and the Runge Phenomenon*, Numerical Methods, pp. 101103 (1974), ISBN 0-13-627315-7.
6. G. Birkhoff and C. De Boor, *Piecewise polynomial interpolation and approximation* in *Approximation of Functions* (H.L. Garabedian, ed.), Elsevier, Amsterdam: 164-190 (1965).
7. P. Bézier, *Example of an existing system in motor industry: The Unisurf system*, Proc. Royal Soc. London, **A321**: 207-218 (1971);
8. A. Forrest, *Interactive interpolation and approximation by Bézier polynomials*, The Computer Journal **15(1)**: 71-79 (1972), reprinted in CAD **22(9)**: 527-537 (1990).
9. G. Farin, J. Hoschek and M.-S. Kim (editors), *Handbook of Computer Aided Geometric Design*, to be published in Elsevier Science, Amsterdam, 2002 (the online manuscript is currently available at: `http://cagd.snu.ac.kr/main.html`).
10. A. Alon and S. Bergmann *Generic Smooth Connection Functions: A New Analytic Approach to Hermite Interpolation* J. Phys. A: Math. Gen. **35** 3877(2002).
11. C. De Boor, K. Höllig and M. Sabin, *High accuracy geometric Hermite interpolation*, CAGD **4**: 269-278 (1987).
12. G. Farin, *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide* (third edition), Academic Press, San Diego (1993).
13. K. Höllig and J. Koch, *Geometric Hermite interpolation.* CAGD **12**: 567-580 (1995).
14. K. Höllig and J. Koch, *Geometric Hermite interpolation with maximal order ang smoothness*, CAGD **13**: 681-695 (1996).
15. X. Lianghong and S. Jianhong, *Geometric Hermite interpolation for space curves*, CAGD **18**: 817-829 (2001).
16. G. Brunnett, H. Hagen and P. Santarelli, *Variational design of curves and surfaces*, Surveys Math. Indust. **3(1)**: 1-27 (1993).
17. W. Wesselink and R.C. Veltkamp, *Interactive design of constrained variational curves*, CAGD **12(5)**: 533-546 (1995).
18. L.A. Pars, *An Introduction to the Calculus of Variations*, Heinemann, London (1962).
19. E. Noether, Nachrichten Gesell. Wissenschaft, Göttingen **2**: 235 (1918).
20. C. L. Mueller and I. F. Sbalzarini. Gaussian Adaptation revisited - an entropic view on Covariance Matrix Adaptation. In Proc. EvoStar, volume 6024 of Lecture Notes in Computer Science, pages 432-441, Istanbul, Turkey, April 2010. Springer.
21. C. L. Mueller and I. F. Sbalzarini. Gaussian Adaptation as a unifying framework for continuous black-box optimization and adaptive Monte Carlo sampling. In Proc. IEEE Congress on Evolutionary Computation (CEC), Barcelona, Spain, July 2010.

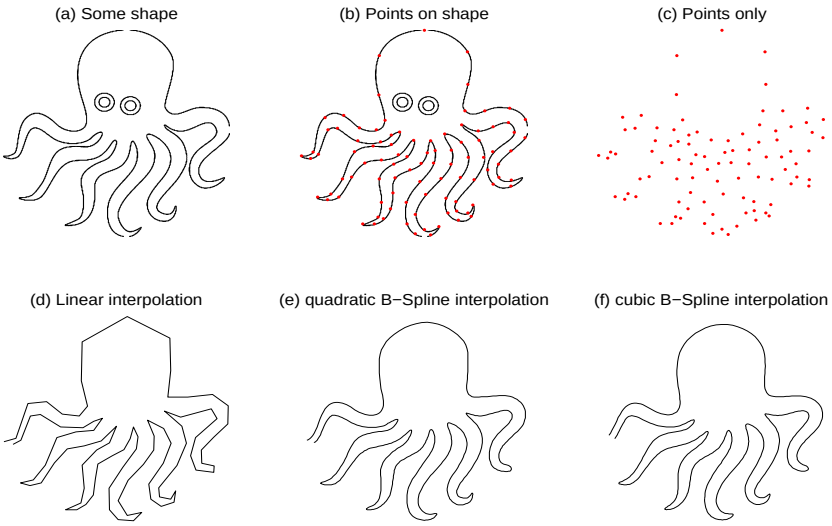

Fig. 1. Example for "Join the dots": Some shape defined by a curve along its boundary (a) can be approximated based on a list of (ordered) points along this curve (b). Having just these dots (c) allows for different ways of reconnecting the dots by interpolation. Linear interpolation (d) induces discontinuities in the slope at each point. Using quadratic (e) or cubic (c) B-splines gives rise to much smoother interpolations, which are almost identical due to the high density of the points.

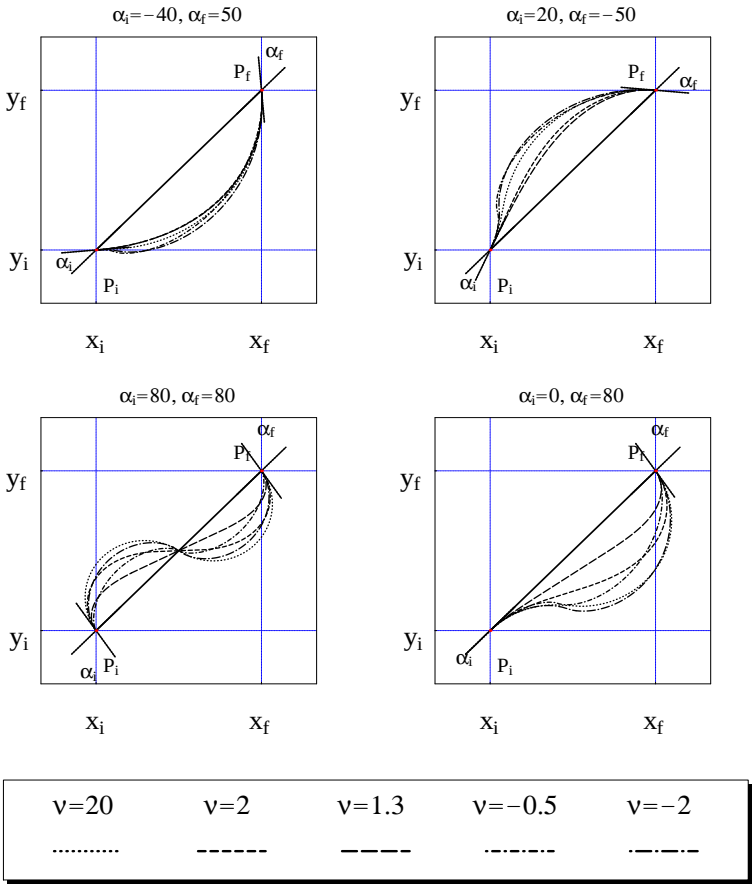

Fig. 2. Examples for "Smooth Connection Functions" that connect the initial point $\boldsymbol{P}_i = (x_i, y_i)$ with the final point $\mathbf{r}_f = (x_f, y_f)$. At $\boldsymbol{P}_i$ and $\boldsymbol{P}_f$ the curves are tangent to the associated rays (dashed) which are specified by their inclination angles, α_i and α_f, with respect to the vector pointing from $\boldsymbol{P}_i$ to $\boldsymbol{P}_f$. Each plot corresponds to a particular choice of (α_i, α_f) and shows the behavior of the solution for five different values of ν as indicated in the legend (in degrees).

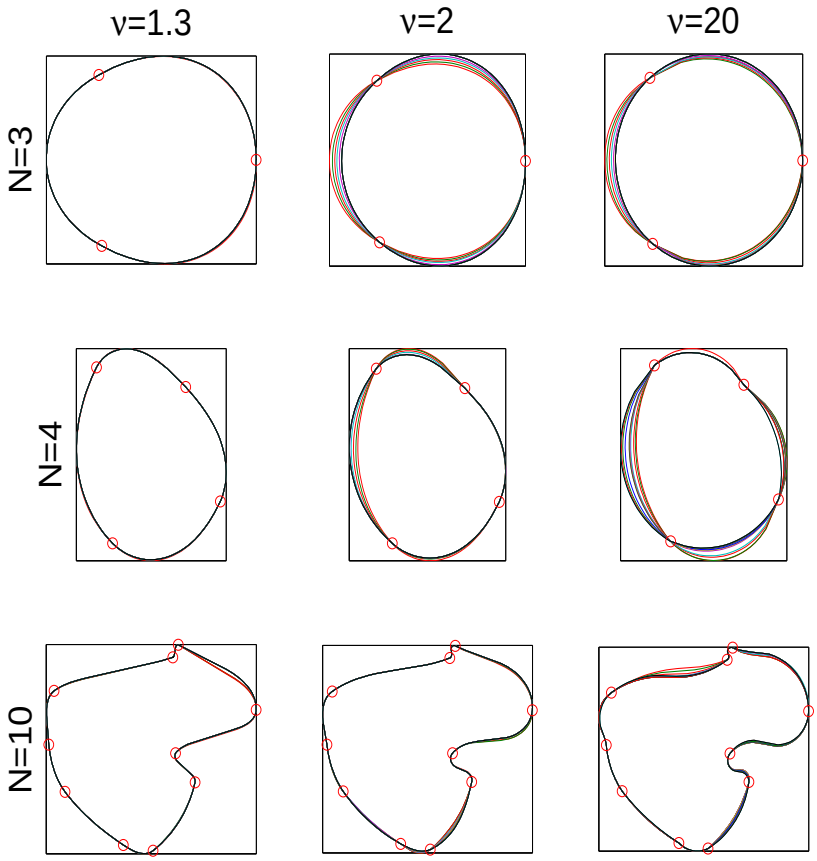

Fig. 3. Examples for interpolation curve for three lists of points and different ν. The evolution of each curve under the iterations of our optimization procedure is indicated by color code. The initial curve is shown in black and the final, optimal curve in red. Intermediate curves are thinner and shown with colors ranging from blue to red. The top panel corresponds two three points, the middle panel to four points, and the bottom panel to 10 points.

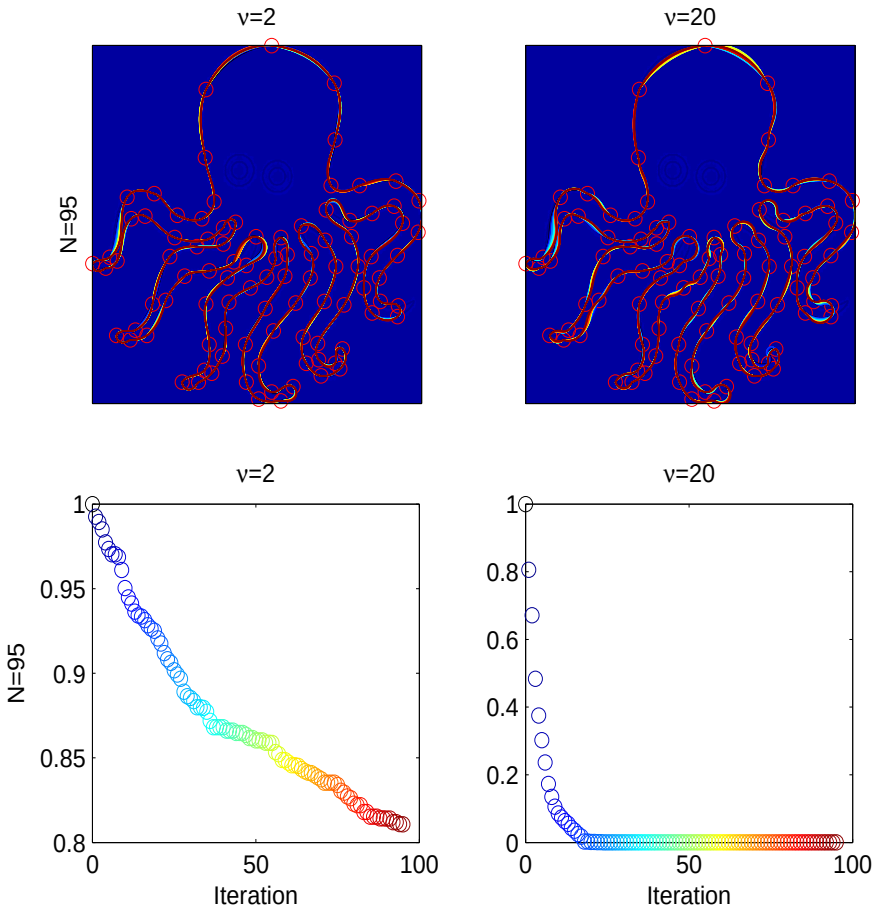

Fig. 4. Examples for interpolation curves for the join-the-dots example with 95 points (see Fig. 1) for $\nu = 2$ (top left panel) and $\nu = 20$ (top right). The evolution of each curve under the iterations of our optimization procedure is indicated by color code. The initial curve is shown in black and the final, optimal curve in red. Intermediate curves are thinner and shown with colors ranging from blue to red. The corresponding action (in units of the action corresponding to the initial guess of the slopes) are shown in the lower panel.

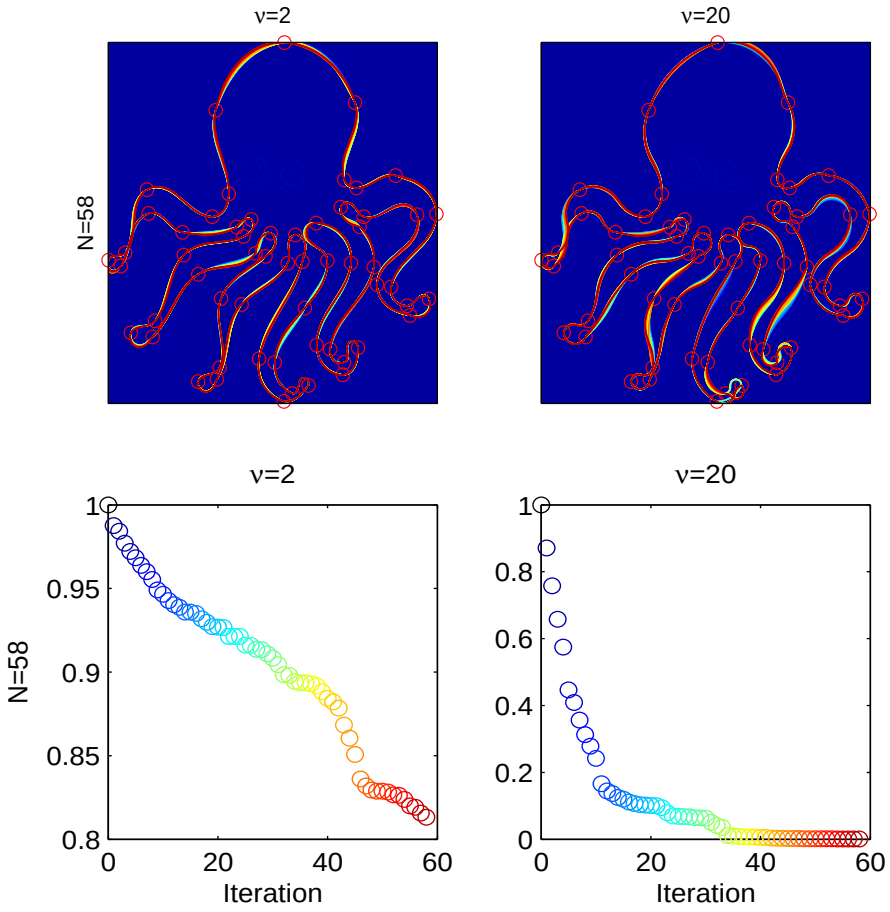

Fig. 5. Same as Fig. 4, but using only 58 dots to define the shape.