EUROGRAPHICS 2015 / O. Sorkine-Hornung and M. Wimmer
(Guest Editors)

Volume 34 (2015), Number 2

Optimal Spline Approximation via ℓ_0-Minimization

Christopher Brandt[1,2] Hans-Peter Seidel[2] Klaus Hildebrandt[1,2]

[1]Delft University of Technology [2]Max Planck Institute for Informatics

Abstract

Splines are part of the standard toolbox for the approximation of functions and curves in $\mathbb{R}^d$. Still, the problem of finding the spline that best approximates an input function or curve is ill-posed, since in general this yields a "spline" with an infinite number of segments. The problem can be regularized by adding a penalty term for the number of spline segments. We show how this idea can be formulated as an ℓ_0-regularized quadratic problem. This gives us a notion of optimal approximating splines that depend on one parameter, which weights the approximation error against the number of segments. We detail this concept for different types of splines including B-splines and composite Bézier curves. Based on the latest development in the field of sparse approximation, we devise a solver for the resulting minimization problems and show applications to spline approximation of planar and space curves and to spline conversion of motion capture data.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Splines

1. Introduction

Splines are widely used in graphics for the approximation of functions and parametrized curves. They combine an efficient representation of "smooth" functions by few parameters and with good approximation properties: under refinement, interpolating splines converge to a (smooth enough) function in various norms. Here we consider the question of *optimal approximating splines* in a general setting, where not only variations of the spline parameter, but also of the number of spline segments and the locations of the knots are allowed. In this setting, the problem of finding the spline that best approximates a smooth function is ill-posed in the sense that by refining the spline, the approximation error can be reduced, and, thus, the limit of a minimizing sequence is not a spline anymore. The problem can be regularized by adding a penalty for the number of spline segments to the objective functional. This means we are looking for a compromise between approximation error and the number of spline segments.

We derive a discrete version of the regularized problem, in which the domain of the spline is uniformly sampled and the feasible set is restricted to splines whose knots are grid points. We show that the discrete problem can be written as an ℓ_0-regularized minimization problem. In this setting, splines are represented by their function values (and depending on the type of spline also their derivatives) at the grid points. The main ingredient to our modeling of the optimization problem is a linear operator on the space of functions on the grid that has the property that the ℓ_0-norm of the image of a function agrees with the number of segments of the corresponding spline, which interpolates the function values. We show how such an operator can be constructed for different families of splines including B-splines, composite Bézier curves, splines in tension, and wiggly splines.

In recent years, the ℓ_0-regularization of optimization problems has received much attention and many efficient algorithms for approximating the solutions have been proposed. Formulating the optimal spline approximation problem as an ℓ_0-minimization problem allows us to use this rapidly growing pool of algorithms for computing optimal splines. Moreover, we see a connection between optimal spline approximation and recent schemes for image and geometry denoising, which yield a similar type of optimization problem. We have tested various algorithms for solving the ℓ_0-regularized optimization problem and propose a variant of the recent scheme by Patrascu and Necoara [PN14]. We tested our implementation for spline approximation of planar and space curves and for spline conversion of motion capture data.

© 2015 The Author(s)
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd. Published by John Wiley & Sons Ltd.

2. Related work

Optimal splines. The computation of optimal approximating splines comes at different complexities depending on what parameters of a spline are varied. Typically, only the parameters the spline depends linearly on are optimized, which leads to linear least-squares problems. For B-splines, this means that a knot vector is fixed and the remaining parameters are optimized. Latest solvers for this problem can compute highly accurate solutions within few milliseconds even for a large number of points [DL14]. However, the approximation can be greatly improved by treating the knots as free variables, see [dB73, Bur74, Jup78] for some early work in this direction. The task is to find an optimal knot vector, in the sense that the best approximating spline on this knot vector yields the lowest approximation error among all other splines with the same number of knots. Strategies to find an optimal knot vector can be roughly classified in three categories:

- The knot vector is iteratively extended using heuristics, such as integrated discrete curvature [LXZG05], largest accumulated $\mathcal{L}_2$ error on a segment [DL14], or others [YWS04, LM88, Vas96]. These methods often require initial knot vectors, user-defined error bounds and other parameters.
- An existing spline gets simplified by iteratively removing knots from the knot vector (*knot removal*), see *e.g.* [LM87]. In each iteration, the knots get ranked via some heuristic criterion and then greedily removed, while maintaining some prescribed proximity to the input curve.
- The knots are regarded as additional free variables (subject to appropriate constraints) and the approximation error of the best spline using these knots is minimized. This, however, leads to a very involved minimization problem (cf. [dB73]), and the latest attempts at solving it use genetic algorithms, which interpret the knot vectors as populations which undergo mutations [ZZYL11, SR01, MYH99, YHY03] or apply particle swarms [GI11].

Our method differs from these approaches in that we are neither limited to B-splines nor is there a need to use explicit geometric constructions or any heuristics in our formulation to find the target set of points which we use to create the spline approximating the input curve. Furthermore, we do not need to fix the number of knots, but keep it variable. By discretizing the parameter interval, we are able to reformulate the optimal spline approximation problem as a ℓ_0-regularized quadratic minimization problem. This opens the door to using recent approximation algorithms for ℓ_0-minimization for optimal spline approximation. For arc splines—curves consisting of a finite sequence of circular arcs and line segments—a geometric approach for approximating a sequence of points by a G^1-continuous arc spline with a minimum number of segments has been recently proposed by Maier [Mai14]. Contemporaneous to our research, Kang et al. [KCL*15] proposed an equivalent reformulation of the optimal spline problem for the case of B-splines in one dimension. However, they replace the ℓ_0-regularized functional by an ℓ_1-regularized functional. We further discuss and compare the ℓ_0- to the ℓ_1-regularization in Section 5.

Optimal polylines. Optimal approximation by polylines has a more local character: changing the position of knots will have no global effect on the solution, as is the case for splines, where differentiability conditions introduce global dependencies. This local property of polylines (and other types of curve which are only demanded to be continuous) allows the use of Dynamic Programming algorithms, which can give optimal approximations (given either a maximum number of segments, a cost per segment or a maximum approximation error in the ℓ_2 or ℓ_∞ norm) by polylines [PV94, GB04], line segments and circular arcs [Kol12, Mai14], or piecewise polynomials [MJEM02].

ℓ_0-Minimization. Sparsity-regularized and constrained convex optimization problems are the focus of recent research, see *e.g.* [PN14, BBR11, BE13]. Though such optimization problems are known to be *NP*-hard, various efficient approximation algorithms have been proposed in recent years, see [BD10, Fou11, XLXJ11, CJPT13, PN14] and references therein. Whereas typically sparsity is demanded in the variable which is minimized, in our case, we want to apply the regularization on a linear transform of the variable. This type of minimization has been considered in recent sparse image processing [XLXJ11, NNZC08] and geometry processing [HS13] applications. The problem has been analyzed and several algorithms have been proposed [CJPT13, XLXJ11, BM98]. We applied, adapted and compared several of these recently proposed algorithms to approach our specific ℓ_0-regularized minimization problem.

3. Optimal spline approximation via ℓ_0-minimization

Our approach for optimal spline approximation can be applied for various types of splines, including B-splines and composite Bézier curves. In this section, we first outline a continuous version of the optimization problem and then introduce the ℓ_0-minimization problem, which is a discrete version of the continuous problem. We explicitly describe the optimization problem for C^2 cubic B-splines and composite cubic Bézier curves. Finally we discuss the generalization to other types of splines.

Continuous problem. Let us consider a parametrized curve $c : I \to \mathbb{R}^d$. Here I is either $[0,1]$ in the case of curves with boundary, or $[0,1]_{/0\sim1}$ (the unit interval with identified boundary) in the case of closed curves. Our goal is to construct a spline $s : I \to \mathbb{R}^d$ that is an optimal trade-off between the approximation of c on the one hand and the number of spline nodes on the other hand (a *node* being an interpolation point together with the corresponding parameter value). For this, we consider the functional

$$\mathcal{E}(s) = \|c - s\|^2_{\mathcal{L}_2} + \lambda\, \nu(s) \tag{1}$$

© 2015 The Author(s)
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd.

(a) $\lambda = 50 \cdot 10^{-9}$, 9 points, ℓ_2-error $\approx 3.80 \cdot 10^{-4}$ (b) $\lambda = 1.05 \cdot 10^{-9}$, 17 points, ℓ_2-error $\approx 7.82 \cdot 10^{-6}$

Figure 1: Cross section of a Beetle, 500 points, approximated using composite cubic Bézier curves generated with our method.

where $\nu(s)$ is the number of nodes of s and $\lambda \in \mathbb{R}_+$. Then an optimal approximating spline is a minimizer of the functional among the set of all splines of one type. The parameter λ provides control of the complexity (number of nodes) of the spline. The optimal spline has the property that there is no spline $\tilde{s}$ with $\nu(\tilde{s}) > \nu(s)$ such that the approximation error $\|c - \tilde{s}\|^2_{\mathcal{L}_2}$ is decreased by more than $(\nu(\tilde{s}) - \nu(s))\lambda$.

B-spline functions. We consider a discretized version of the optimization problem. For simplicity of presentation, we describe the case of C^2 cubic B-spline approximation of functions first, and generalize to other types of splines and the approximation of curves in $\mathbb{R}^d$ later.

Let $t_1 < t_2 < \cdots < t_N$ be a dense uniform partition of the interval I. The data we want to approximate is a set of N function values $p_i \in \mathbb{R}$, with corresponding parameter values $t_i \in I$, *e.g.*, a sampling of the continuous input function. A C^2 cubic B-spline can be characterized as the unique (C^2-continuous) cubic spline that interpolates a set of $\tilde{N}$ nodes $(\tilde{t}_i, q_i) \in I \times \mathbb{R}$ and satisfies certain boundary conditions. Computing the corresponding control points amounts to solving a linear system of equations. Here, we will represent a cubic spline by the choice of the interpolating points and times. To be able to optimize over a set of cubic splines with a variable number of segments and variable nodes, we restrict the $\tilde{t}_i$ to be values of the aforementioned partition of I. With this restriction, any cubic spline s can be represented by the vector $q \in \mathbb{R}^N$ listing the images $q_i = s(t_i) \in \mathbb{R}$ for all t_i. For any q, we call those q_i that are not nodes of the spline corresponding to q the *inner points* of q.

The inner points can be characterized as follows: Let $s(t)$ be a cubic spline, interpolating the nodes (t_i, q_i), *i.e.* $s(t_i) = q_i$, with some prescribed boundary conditions at t_1 and t_N, and now assume that $s'''(t)$ is continuous at $t = t_j$ for some $j \in 2, \ldots, N-1$. Then the cubic polynomials $s(t)|_{t \in [t_{j-1}, t_j]}$ and $s(t)|_{t \in [t_j, t_{j+1}]}$ are the same, since the values and all three derivatives at $t = t_j$ agree. This means that the cubic splines interpolating the nodes $\{(t_1, q_1), \ldots (t_1, q_N)\}$ and $\{(t_1, q_1), \ldots (t_1, q_N)\} \setminus (t_j, q_j)$ are the same. Conversely, if for some t the third derivative s''' has a discontinuity, then $s(t)$ must be a node of s. So we have that q_j is an inner point if, and only if, the third derivatives of s from the left and from the right of t_j agree. This condition is linear in the coordinates representing the spline, so there is a linear operator $C : \mathbb{R}^N \to \mathbb{R}^N$ with the property that q_j is an inner point, if and only if $(Cq)_j = 0$. The operator C depends on the choice of boundary conditions. In the appendix, we discuss the construction of this operator explicitly. Given a vector $q \in \mathbb{R}^N$, we can construct the resulting cubic spline by interpolating all nodes of q (those for which $(Cq)_j \neq 0$) by a cubic spline s with appropriate boundary conditions. Then, all inner points of q will lie on this resulting spline.

The number $\nu(s)$ of nodes of the cubic spline s corresponding to a vector $q \in \mathbb{R}^N$ agrees with the ℓ_0-norm of Cq. Our discrete analog of the energy (1) is

$$E(q) = \|p - q\|^2_{\ell_2} + \lambda \|Cq\|_{\ell_0}, \tag{2}$$

where $p \in \mathbb{R}^N$ lists the input data points. Computing an optimal approximating spline amounts to minimizing (2) over all $q \in \mathbb{R}^N$ and computing the cubic B-spline s corresponding to the minimizer q. In the case that the interval is $I = [0, 1]$, one can additionally enforce interpolation of the boundary, *i.e.*, $q_1 = p_1$ and $q_N = p_N$. Boundary conditions on the derivatives are incorporated into the operator C and the reconstruction of the spline s from a vector q.

B-splines curves. To approximate curves in $\mathbb{R}^d$, our construction remains almost unchanged: the curve samples p_i and our spline representation q_i are elements of $\mathbb{R}^d$ and we write the list of points p and q as $N \times d$ matrices. This way, the energy (2) remains the same except that the ℓ_2 norm gets replaced by the Frobenius norm $\|\cdot\|_F$, and the ℓ_0 term is interpreted row-wise, *i.e.*, $\|Cq\|_{\ell_0}$ counts the number of rows of $Cq \in \mathbb{R}^{N \times d}$ which contain non-zero entries.

Composite cubic Bézier curves. For composite cubic Bézier curves (*CCBC*), the derivation is similar to that of B-splines. These curves are composed of cubic polynomials, but instead of demanding the second derivative to be continuous, first derivatives at all nodes can be prescribed. As for the B-splines, we represent CCBCs with a variable number of nodes, by storing $q_i = s(t_i)$ for all t_i of the uniform partition of I. However, for representing the CCBCs, we additionally store the derivatives $q'_j = s'(t_j)$. Then, any CCBC whose nodes are a subset of the partition $t_1, t_2, \ldots, t_N$ can be reconstructed.

For initialization, the q'_is can be estimated from the input data p or explicitly computed if the input is a parametrized curve. Then, we search for an optimized set of points

© 2015 The Author(s)
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd.

and derivatives (q, q'), which means that the conditions for points to be inner points are posed on q *and* q'. The characterization of the inner points for CCBCs is similar to that for B-splines. However, in contrast to the case of B-splines, the second derivative is not guaranteed to be continuous, and thus we have to pose two conditions on inner points, namely that the second and third derivatives are continuous at the corresponding t_i. Again, these conditions are linear and can be formulated via a linear operator $C : \mathbb{R}^{2N} \to \mathbb{R}^{2N}$, such that p_j is an inner point, if $(C(p,p'))_k = 0$ and $(C(p,p'))_l = 0$, where k and l are the indices of the rows in $C(p,p')$ which are associated to the condition that the second and third derivatives are continuous at t_j. The construction of the operator is discussed in more detail in the appendix. Using it, we can formulate the energy associated to CCBCs as

$$E\left(q,q'\right) = \|p-q\|_F^2 + \lambda \left\|C(q,q')\right\|_{\ell_0}, \tag{3}$$

where the ℓ_0 term is now counting the number of points for which one or more of the $2 \cdot d$ conditions are not satisfied. To compute an optimal CCBC, we minimize the energy (3) over all q and q' and reconstruct the CCBC from this data. Boundary conditions can be enforced by imposing equality constraints onto q_1, q_N, q_1', and q_N'.

Other spline types. B-splines and composite Bézier curves of higher order can be treated in the same vein as above, by introducing more conditions per point and potentially additional variables.

More types of splines can be covered by the following construction: suppose, that the interpolating spline through the nodes (q_i, t_i) is the minimizer of some energy $E : C^k([0,1]) \to \mathbb{R}$, subject to the interpolation constraints. For example, cubic splines are minimizers of the quadratic functional $F(f) = \int_0^1 \left\|f''(t)\right\|^2$, $f \in C^2$, subject to $f(t_i) = q_i$. Furthermore let $s_q : [0,1] \to \mathbb{R}^d$ be the interpolating spline through the points q (for some fixed parameter values t_i) and $\hat{E}(q) = E(s_q)$. Then the gradient mapping of $\hat{E}$ can be used as the operator C, since if the gradient is 0 at a point, the interpolation constraint of this point can be left out, yielding the same minimizer. Thus it is an inner point. Applying this construction to cubic splines, yields the same matrix C as constructed in the appendix. Another example for such a spline type are splines in tension [Sch66, HP04], which are minimizers of the quadratic functional $\int_0^1 \left(f''(t)\right)^2 + \mu \int_0^1 \left(f'(t)\right)^2$, $f \in C^2$, subject to $f(t_i) = q_i$. In Figure 5, we demonstrate the effect of varying μ and using our method to produce optimal approximating splines in tension. Wiggly splines [KA08, HSvTP12, SvTSH14] interpolating the nodes (q_i, t_i) are minimizers of $\int_0^1 \left(f''(t) + \delta f'(t) + \xi f(t)\right)^2$, $f \in C^2$, subject to $f(t_i) = q_i$, and thus there is a corresponding matrix C characterizing inner points for wiggly splines.

To construct the matrix C for these types of splines, one first needs to find the matrix A which maps the point coordinates to the coefficients of the interpolating spline. This matrix will depend on the t_i, the parameters (μ for splines in tension and δ, ξ for wiggly splines) and possible boundary conditions. Since the energy characterizing the splines

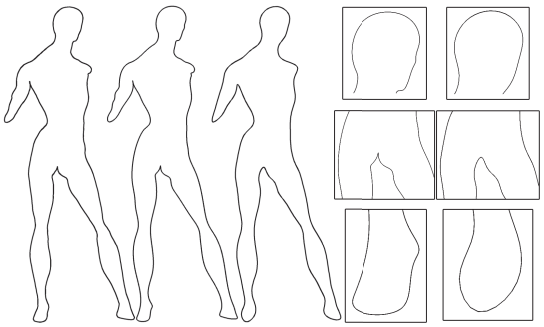


Figure 2: Comparison of our method to equidistant interpolation points. Left: input (500 points), middle: our method, cubic spline through 100 points, right: cubic spline through 100 equidistant points. On the right side are magnified regions of our result (left boxes) and the equidistant result (right boxes).

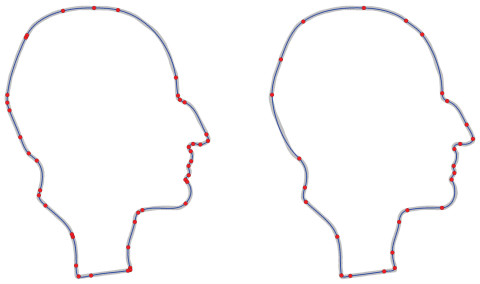


Figure 3: Comparison of our approximation method for cubic B-splines (left, 42 points) and CCBCs (right, 27 points) on the silhouette of a bust of Max Planck, 300 points, $\lambda = 4 \cdot 10^{-9}$ for both spline types.

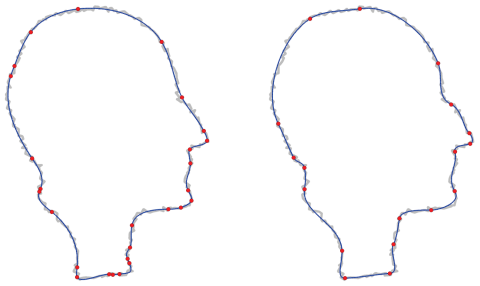


Figure 4: The silhouette of a bust of Max Planck with extreme noise (300 points) approximated by a cubic B-spline (left) and a CCBC (right) using our method with $\lambda = 60 \cdot 10^{-9}$.

© 2015 The Author(s)
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd.

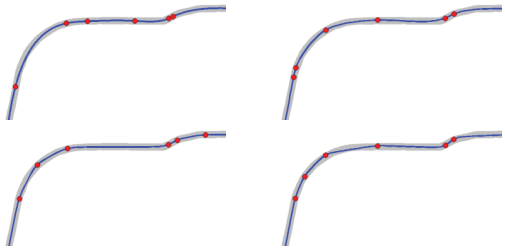


Figure 5: Testing our method on splines in tension. In reading order, we set $\mu = 0$, 0.1, 0.5 and 1. $\lambda = 0.5 \cdot 10^{-9}$ was used in all cases.

is quadratic, it comes from a linear system of equations describing the relation between interpolation points and spline coefficients. Then we get $C = A^T BA$, where B maps the spline coefficients c to the value of the energy via $c^T Bc$.

4. Numerical optimization of ℓ_0-regularized problems

In recent years, many approximation algorithms for ℓ_0-regularized problems have been developed. The classical ℓ_0-regularized optimization problem is

$$q^{\text{opt}} = \arg\min_q f(q) + \lambda \|q\|_{\ell_0} \tag{4}$$

where $f(q)$ is a convex function [PN14, BBR11, BE13]. In our case, however, we want to find solutions where Cq is sparse, *i.e.*,

$$q^{\text{opt}} = \arg\min_q f(q) + \lambda \|Cq\|_{\ell_0} \tag{5}$$

where $f(q)$ is the squared ℓ_2 distance of q to the input points p, and C is a linear operator which might not be invertible—as, for example, is the case for the matrices resulting for cubic B-splines and CCBCs described in the previous sections—so algorithms which solve equation (4) cannot be applied directly.

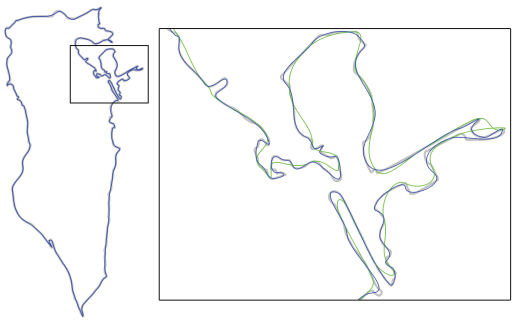


Figure 6: Comparison of a CCBC generated by our algorithm to a CCBC on equidistant nodes on the outline of Bahrain (2000 points).

Luckily, energy functionals of the form (5) have received much attention in the graphics and vision community, where they are used for sparse image recovery and smoothing [CJPT13, XLXJ11, NNZC08] and mesh denoising [HS13].

We implemented and compared several algorithms, including

- a method proposed by Xu et al. [XLXJ11] (ℓ_0-*gradient-minimization*), where an auxiliary variable is introduced and minimization is alternated between two modified versions of (5);
- a majorize-minimize subspace algorithm with ℓ_2–ℓ_0 regularization [CJPT13], where the ℓ_0 term is replaced by a differentiable approximation term, and the functional is then minimized using a subspace gradient-descent method with a majorize-minimize step size search;
- a slightly modified hard-thresholding pursuit algorithm inspired by [Fou11, BD10], where we performed a change of variables $\tilde{q} = Cq$, computed the gradient of $f(C^{-1}\tilde{q})$ by interpreting $C^{-1}\tilde{q}$ as a least-squares solution of $\tilde{q} = Cq$ subject to $\sum q_i = 1$, and then finding the optimal spline through the points with the largest gradient in these coordinates;
- a greedy matching pursuit algorithm [BM98], where we iteratively add a small group of points as nodes (by removing the corresponding conditions $(Cq)_i$ from a list of all constraints), which was best (in terms of the ℓ_2 error) among a random selection of point groups;
- a modified random coordinate-descent method inspired by [PN14], which we will cover in more detail below;

For most experiments conducted in Section 5, the random coordinate-descent algorithm yielded the lowest values of the energy functional. The algorithms that do not directly control which entries of Cq are set to 0 (the first two listed) are problematic, since it is difficult to decide which points to use as nodes in the end.

Thus, our proposed solver uses a modified version of the method introduced by Patrascu and Necoara [PN14], who aim at minimizing (4) by a random coordinate-descent method. Our modified algorithm—aimed to minimize (5)—is summarized in Algorithm 1. We avoid a change of variables, since for our matrices C this results in an unstable algorithm. Instead we transform the minimization step in each iteration into solving a linearly constrained quadratic program.

In our final implementation, we choose the random points from a shuffled list of all points and stop after all points have been visited once. (As an optional last step, one can revisit all still unconstrained points again and check if they can be constrained without increasing the energy by more than λ.) This means that we have N steps, and in each step a quadratic functional has to be minimized subject to at most $2Nd$ linear equality constraints. The method is straightforward to implement, and the constrained minimization requires only solving a linear system. The quadratic programs

© 2015 The Author(s)
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd.

Data: Set of points p, sampling of the input curve.
Result: A minimizer p^{opt} of (5)
$L \longleftarrow$ empty list of indices of vertices to be constrained;
$\hat{p} \longleftarrow p$;
repeat
  Choose an index $k \in \{1,...,N\} \setminus L$;
  $\hat{L} \longleftarrow L$;
  $L \longleftarrow L \cup \{k\}$;
  Solve $p^{\text{opt}} = \arg\min_q \|q-p\|^2$ subject to the constraints $(Cq)_k = (0,...,0) \in \mathbb{R}^d$ for all $k \in L$;
  if $\|p^{opt}-p\|^2 - \|\hat{p}-p\|^2 < \lambda$ **then**
    $\hat{p} \longleftarrow p^{\text{opt}}$;
  else
    $L \longleftarrow \hat{L}$;
  end
until *convergence*;

Algorithm 1: Our modified random coordinate-descent method for ℓ_0-regularized optimization

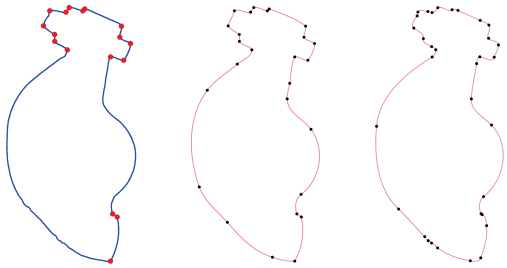


Figure 7: A cross section of the rocker-arm model with 500 points and 17 points marked as discontinuities in the first derivative (left). Optimized composite cubic Bézier curves with (middle, 26 points) and without (right, 33 points) these discontinuities, for the same value of λ.

to be solved in each iteration are related since at most one equality constraint is added per iteration. Therefore, instead of solving the whole problem in every iteration, information from the previous iteration can be used to speed up the computation (*e.g.* updating of the matrix factorization of the previous iteration instead of computing a new factorization). Such procedures are provided by various optimization libraries—we used MOSEK [AA00]. The average times for solving the quadratic programs in different scenarios are listed in Table 3. Minimizing the ℓ_0-regularized energy via a random coordinate-descent can be regarded as a variant of traditional knot removal techniques. However, this relation to knot removal techniques is specific to the random coordinate-descent solver. In this sense, the ℓ_0-formulation we introduce opens a door for using solvers like matching pursuit or ℓ_0-gradient minimization for knot removal. This could be particularly helpful when removing knots from a spline with a large number of knots since the number of iterations required by the random coordinate-descent (as well as knot removal techniques) depends on the number of samples of the input curve (knots of the input spline). In this case, other solvers for the ℓ_0 problem (*e.g.* ℓ_0-gradient-minimization), for which the required number of iterations does not directly depend on the number of sample points, are an alternative.

5. Experiments and comparisons

In our experiments, we scaled the curves to have unit arc length and sampled them equidistantly, densely enough to preserve the details of the input curve. All relevant values of the depicted experiments can be found in Table 1. Additionally, in Figure 10, we plot the ℓ_2 error and the resulting number of points of the result of our method (using cubic splines) against various values of λ. From this, it can be seen that λ determines the degree of detail in the approximating curve as expected: the higher we set the cost per segment, the fewer points we get, at the cost of higher ℓ_2 errors.

Figures 1 and 3 show how our method smartly places points in order to reduce the ℓ_2 error. Figure 1 shows the effect of decreasing λ: as the cost per segment becomes lower, more segments are placed in favor of improving the approximation of the input curve. We found that $\lambda = 10^{-9}$ gave a good balance between the number of points and accuracy of the approximation for curves of unit arc length, but of course the value has to be adjusted for specific tasks and depending on the demands of applications. While it is possible to adjust the level of detail via controlling the desired number of nodes, we found that it is more intuitive to set a cost per segment, since this is a measure that transports over all curves of the same total length. Figure 3 compares cubic splines to CCBCs when approximating a curve using our method with the same value of λ. Note that CCBCs need considerably fewer nodes to achieve the same level of detail. However, at each node we specify the position as well as the first derivative of the spline, instead of just the position.

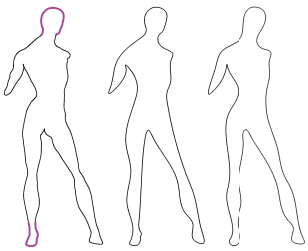


Figure 8: Results of our method with weighted ℓ_2 norm. Left: input with marked points in purple, middle: cubic spline through 49 points using weighted ℓ_2 (weight 100 at the marked vertices, 1 elsewhere), right: cubic spline from 49 points using standard ℓ_2.

© 2015 The Author(s)
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd.

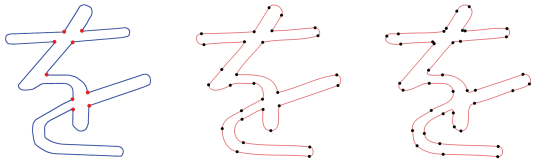


Figure 9: The Hiragana "wo" with 500 points and 9 points marked as discontinuities in the first derivative (left). Composite cubic Bézier curves received from our algorithm with (middle, 29 points) and without (right, 39 points) these discontinuities. Both approximations have similar ℓ_∞ error.

Our method is robust to noise: one can adjust λ such that it captures all desired details but not the noise, and the interpolated points will be chosen and arranged such that a smooth curve with minimal ℓ_2 distance is achieved. This is demonstrated in Figure 4 for both CCBCs and cubic B-splines: no noise is visible in the results while most of the curve's characteristics are still intact.

Figures 7 and 9 show the flexibility of our method: by simply modifying the constraints of certain points (which means changes to the operator C), we are able to specify points at which we allow the CCBC to have discontinuities in the first derivative, which is very useful if sharp edges need to be modeled, as in these examples. The results are compared to the case where no discontinuities in the first derivative are allowed. In Figure 8, we used a weighted ℓ_2 norm: some points were assigned a higher weight such that in the resulting cubic spline we captured the details in these parts very closely, while saving points by approximating the rest only roughly.

In Figure 12, we test our algorithm on a large set of three-dimensional densely sampled curves which represent smooth feature lines on a scanned mesh. This example also shows the advantage of the l_0-regularized minimization: instead of having to choose a fixed number of points for every curve on the mesh, we fix the parameter λ once and obtain a comparable level of detail across all resulting splines.

As another application, we applied our algorithm to motion-capturing data. The data describes the motion of a human performing capoeira over 400 frames (13.3 seconds at 30 frames per second). It contains the position of the midpoint and 3-dimensional angles of 28 joints between various parts of the body, resulting in 89D data (cf. Figure 13). First, we approximated the data using one CCBC for the position and one for each angle, resulting in 29 3-dimensional splines. We chose λ such that the approximating splines had 25 nodes on average, which led to an approximation error of $6.51 \cdot 10^{-4}$ (the data was normalized to have unit arc length as well). Visually, the animation generated by the approximating spline is almost indistinguishable from the input animation, which can be seen in the supplementary video. Additionally, we used our algorithm to smooth a part of the animation in which the human was visibly trembling: after approximating 100 frames through one 89-dimensional CCBC with 20 nodes, the primary motion of the human was still completely intact while all trembling was gone.

Name (Fig.)	Type	O	R	$\lambda \cdot 10^9$	ℓ_2-error	ℓ_∞-error
Beetle (1a)	CCBC	500	9	50	$3.80 \cdot 10^{-4}$	0.00288
Beetle (1b)	CCBC	500	17	1.05	$7.82 \cdot 10^{-6}$	0.00289
Max-Planck (3, l)	Cubic	300	42	4	$3.87 \cdot 10^{-5}$	0.00318
Max-Planck (3, r)	CCBC	300	27	4	$2.82 \cdot 10^{-5}$	0.00383
Silhouette (8, m)	Cubic*	500	49	18	$6.75 \cdot 10^{-5}$	0.00510
Silhouette (8, r)	Cubic	500	49	5	$4.22 \cdot 10^{-5}$	0.00758
Noisy Max-Planck (4, l)	CCBC	300	18	60	$2.11 \cdot 10^{-4}$	0.00759
Noisy Max-Planck (4, r)	Cubic	300	26	60	$2.29 \cdot 10^{-4}$	0.00759
Hiragana "wo" (9, m)	CCBC**	500	29	1.5	$8.34 \cdot 10^{-5}$	0.00448
Hiragana "wo" (9, r)	CCBC	500	39	4	$3.50 \cdot 10^{-5}$	0.00446
Rocker-Arm (7, m)	CCBC**	500	26	2	$6.11 \cdot 10^{-5}$	0.00151
Rocker-Arm (7, m)	CCBC	500	33	2	$2.04 \cdot 10^{-5}$	0.00237
3D Feature-lines (12)	CCBC	2033	154	1000	$6.24 \cdot 10^{-6}$	0.00251

O = # pts of input curve, R = # knots of spline, *=weighted ℓ_2, **=w/ disc. in 1st deriv.

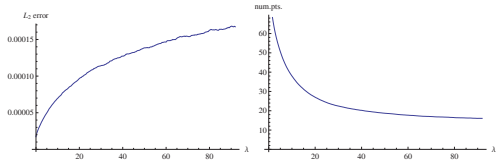
Table 1: Data for the experiments.

	CCBC	Cubic	Cubic w/ lumped mass matrix
250 pts.	3.2	10.4	2.0
500 pts.	7.4	33.6	4.6
1000 pts.	20.1	79.4	11.9

Table 3: Average times (in milliseconds) for solving the quadratic program in each iteration of Algorithm 1

Experimental comparisons. We compare the performance of different solvers for the ℓ_0-regularized optimization problem (2). In addition, we compare the results of the ℓ_0-minimization to results produced by alternative (recent and traditional) techniques for knot optimization. For a set of input curves, we produce cubic splines with the same number of knots using all the methods and measure the resulting ℓ_2-errors. The results are listed in Table 2. In addition to Algorithm 1, we used the following methods:

- A matching pursuit algorithm, where instead of iteratively removing knots, we build the set of unconstrained nodes by greedily unconstraining the best (in terms of lowest resulting energy functional value) cardinality k set, among m randomly chosen cardinality k sets. For the comparisons we used $m = 500, k = 3$ throughout. Note that evaluating the candidate sets in each iteration takes as long as



(a) ℓ_2 errors for various values of λ (b) Number of points for various values of λ

Figure 10: Results of using our method on a 150-point curve for $\lambda \in [0, 100 \cdot 10^{-9}]$, for cubic splines.

© 2015 The Author(s)
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd.

			ℓ_0 optimizer				Other techniques		
Input curve	#pts	#knots	Algorithm 1	Match.Purs.	ℓ_0-gradient-min.	ℓ_1-reg.	PSO [GI11]	[DL14]	MATLAB OPTKNT
Max Planck (Fig. 3)	300	48	$\mathbf{2.7587 \cdot 10^{-5}}$	$3.6802 \cdot 10^{-5}$	$2.8661 \cdot 10^{-5}$	$5.4789 \cdot 10^{-5}$	$4.4474 \cdot 10^{-5}$	$3.6980 \cdot 10^{-5}$	$6.6168 \cdot 10^{-5}$
Cross Section (Fig. 11)	500	48	$1.8336 \cdot 10^{-5}$	$3.3407 \cdot 10^{-5}$	$\mathbf{1.3729 \cdot 10^{-5}}$	$4.9466 \cdot 10^{-5}$	$4.2251 \cdot 10^{-5}$	$3.6252 \cdot 10^{-5}$	$6.1692 \cdot 10^{-5}$
Silhouette (Fig. 8)	500	97	$1.2066 \cdot 10^{-5}$	$1.3967 \cdot 10^{-5}$	$\mathbf{1.0834 \cdot 10^{-5}}$	$1.2337 \cdot 10^{-5}$	$1.9649 \cdot 10^{-5}$	$1.2797 \cdot 10^{-5}$	$2.5114 \cdot 10^{-5}$
Bahrain (Fig. 6)	2000	137	$\mathbf{1.2070 \cdot 10^{-5}}$	$1.6341 \cdot 10^{-5}$	$1.5947 \cdot 10^{-5}$	$3.0374 \cdot 10^{-5}$	$1.5633 \cdot 10^{-5}$	$1.2395 \cdot 10^{-5}$	$1.7437 \cdot 10^{-5}$
Hiragana (Fig. 9)	500	52	$5.6206 \cdot 10^{-5}$	$\mathbf{5.3876 \cdot 10^{-5}}$	$6.9553 \cdot 10^{-5}$	$8.3407 \cdot 10^{-5}$	$5.7682 \cdot 10^{-5}$	$6.1827 \cdot 10^{-5}$	$7.2012 \cdot 10^{-5}$

Table 2: Data for comparisons of different ℓ_0-minimizers and other spline optimization techniques. For a set of input curves, the ℓ_2-errors to the resulting cubic splines are listed. In every row, all splines have the same number of knots.

m full iterations in Algorithm 1, which is why this method is only feasible for a small number of desired knots.

- We minimize energy (2) using the aforementioned ℓ_0 gradient minimization method proposed by Xu et al. [XLXJ11]. The direct results of this method often yield large clusters of nodes, which we address by using a cleanup step after the optimization: we check, for each remaining unconstrained point, whether removing the knot improves the energy. In many cases, the resulting optimizer performs as well as (or sometimes better than) Alg. 1. However, it depends on various parameters that need to be adjusted for each curve.
- When replacing the ℓ_0-term in energy (2) by an ℓ_1 term (as proposed in [KCL*15] for the 1-dim. splines), the resulting convex energy can be minimized directly by solving a quadratic program. However, in this formulation, λ can no longer be interpreted as the cost per segment, but it needs to be experimentally adjusted to achieve the desired level of detail. Furthermore, large values in Cp get punished, which is not desired and leads to over-smoothing of the resulting curve. Also, there is no clear way to couple the constraints for the coordinates of the same node, which typically leads to clusters of unconstrained points. These problems can cause bad performance, an example is the large error for the approximation of the Bahrain-curve.
- For the case of B-splines, our ℓ_0 problem is addressing the free-knot optimization problem, so it is natural to compare to recent techniques which aim at directly minimizing the ℓ_2-error over a fixed number of nodes. Galvez et al. [GI11] report that this results in a difficult multimodal and multivariate nonlinear optimization problem, and they try to find an optimal knot vector by using particle-swarm optimization (PSO). We apply their algorithm using the proposed parameter values, namely 100 particles, 10 iterations, $\gamma_{1,2} = 2$, $w = 0.9...0.4$. While the PSO performs well if the number of desired knots is small, it becomes infeasible for a larger number of knots.
- The refinement algorithm proposed in [DL14] that iteratively adds knots to an approximating cubic B-spline (starting from a cubic B-spline defined on a sparse knot vector) by placing them on the segment of the spline on which the ℓ_2 error is largest, specifically at the parameter value, where the accumulated ℓ_2 error reaches half of the total error on this segment. While performing comparable to our algorithm on curves with fairly constant level of detail, it often produces knots at unnecessary places (see Figure 11) for other types of curves.
- As a comparison to a traditional technique, we consult the MATLAB function OPTKNT which computes an "optimal" knot sequence "in the sense of Micchelli/Rivlin/Winograd and Gaffney/Powell" [GP76].

For all tested examples, one of the ℓ_0-optimizers yielded the best result, which justifies our reformulation of the optimal spline problem.

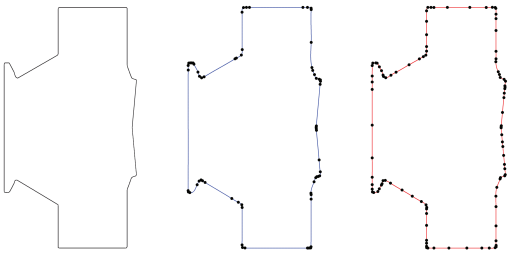


Figure 11: A curve approximated by a cubic spline using our algorithm and using a refinement technique from [DL14], such that both results produce similar ℓ_2 error to the input curve. From left to right: input curve, cubic B-spline from our algorithm with 65 nodes, cubic B-spline from the refinement method in [DL14] with 102 nodes.

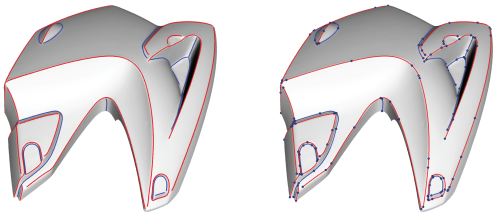


Figure 12: Feature lines on a surface, left: input feature lines, right: optimized CCBCs with 154 vertices.

6. Conclusion

We introduce a new approach for computing optimal approximating splines, where spline coefficients, positions of nodes, and the number of spline segments are variable. The approach can be used to optimize different types of splines

© 2015 The Author(s)
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd.

including B-splines and composite Bézier curves. Our modeling of the optimization problem yields a ℓ_0-regularized quadratic problem for which we devise a solver based on the recent scheme of Patrascu and Necoara [PN14]. We present results produced with our implementation for the approximation of planar and space curves, and spline conversion of motion capture data.

Limitations and challenges. One direction of future work is to find a convex approximation of problem (2), *e.g.* using weighted ℓ_1-norms. This could potentially lead to a very fast tool for optimal spline approximation. A limitation of our current approach is that for B-splines, it cannot deal with multiple knots. One idea of how to integrate multiple knots is to allow for lower regularity than C^2 (for the case of cubic splines) at nodes, but to penalize lower continuity. We leave this problem as future work. Our tool could be used to convert curves (*e.g.* hand-drawn outlines of shapes) into CCBCs (or other splines types), which can be used for curved editing. It would be interesting to experiment with other norms for the approximation of the input curve. The ultimate goal would be to design a norm such that our tool places the nodes in locations where an artists would place them. Another direction for future work is to extend the approach from curves to surfaces.

Appendix A: Explicit Constraint Operators

Cubic splines. We will now construct the operator C for cubic splines, which will have the property that $(C(q))_j = 0$ if the point q_j is an inner point. To this end, we need the operator A which maps a set of points to some representation of a cubic spline. We represent the cubic spline by its second derivatives q_j'' at the points q_j (which determine the spline uniquely), since in this representation all involved operators have simple, precomputable forms. Therefore we define the stiffness matrix S as the circulant matrix having $\frac{h}{6}(2,-1,0,\dots,0,-1) \in \mathbb{R}^N$ as its first row and the mass matrix M as the circulant matrix having $\frac{1}{h}(4,1,0,...,0,1) \in \mathbb{R}^N$ as its first row. Then, determining the second derivatives $q'' = (q_1'',...,q_N'')$ at the points can be done via solving the system of linear equations $Mq'' = -Sq$. This system assumes that the t_i are equidistant with step size h, *i.e.* $t_i - t_{i-1} = h$ for $i = 2,...,N$, and that the curve is closed, *i.e.* we want periodic boundary conditions (other boundary conditions can be realized by changing the first and last rows of the matrices M and S). The matrix $A = \frac{6}{h^2}M^{-1}S$ will also be circulant and its entries can be precomputed up to arbitrary precision. Now the cubic spline for $t \in [t_j, t_{j+1})$ is given by

$$s(t) = q_j s_1(t) + q_{j+1} s_2(t) + \frac{h^2}{6}(q_j''(s_1^3(t) - s_1(t)) + q_{j+1}''(s_2^3(t) - s_2(t)))$$

$$\text{where } s_1(t) = \frac{t_{j+1} - t}{h} \text{ and } s_2(t) = \frac{t - t_j}{h}$$

So for $t \in [t_j, t_{j+1})$ the third derivative of the resulting cubic spline is $s'''(t) = \frac{1}{h}\left(-q_j'' + q_{j+1}''\right)$ and as the condition for the left and right derivatives at $t = t_j$ to agree we get $2q_j'' - q_{j-1}'' - q_{j+1}'' = 0$. Thus, letting $C = SM^{-1}S$, we get the desired linear operator, which has the property that q_j is an inner point iff $(C\,p)_j = 0$.

CCBCs. In order to define the operator C which characterizes inner points for CCBCs, we are going to represent CCBCs by their coefficients in the Bernstein basis $(B_k^3)_{k=0,\dots,3}$, since this again leads to simple forms of the involved operators. It is $B_k^3(t) = \binom{3}{k} t^k (1-t)^{3-k}$ and in case of equidistant sampling with step size h, the CCBC takes the piecewise defined form $s(t) = s_j(t) = \sum_{k=0}^3 b_k^j B_k^3\left(\frac{t-t_j}{h}\right)$ for $t \in [t_j, t_{j+1})$. The coefficients are directly given via the set of points and the prescribed first derivatives (we again assume equidistant step sizes h):

$$b_0^j = q_j, \quad b_1^j = q_j + \frac{h}{3}q_j', \quad b_2^j = q_{j+1} - \frac{h}{3}q_{j+1}', \quad b_3^j = q_{j+1}$$

and the second and third derivatives of $s_j(t)$ for $t = t_j$ and $t = t_{j+1}$ are given by

$$s_j''(t_j) = \frac{6}{h^2}(b_2^j - 2b_1^j + b_0^j), \quad s_j''(t_{j+1}) = \frac{6}{h^2}(b_3^j - 2b_2^j + b_1^j)$$

$$s_j'''(t_j) = s_j'''(t_{j+1}) = \frac{6}{h^3}(b_3^j - 3b_2^j + 3b_1^j - b_0^j)$$

An inner point q_j has to satisfy the conditions $s_j''(t_j) = s_{j-1}''(t_j)$ and $s_j'''(t_j) = s_{j-1}'''(t_j)$. From this, for $n = 1$, we define C as the $2N \times 2N$ matrix

$$\begin{pmatrix}
0 & -\frac{4h}{3} & 1 & -\frac{h}{3} & 0 & \cdots & & & & & \cdots & 0 & -1 & -\frac{h}{3} \\
4 & 0 & -2 & h & 0 & \cdots & & & & & \cdots & 0 & -4 & -h \\
-1 & -\frac{h}{3} & 0 & -\frac{4h}{3} & 1 & -\frac{h}{3} & 0 & \cdots & & & & & \cdots & 0 \\
-2 & -h & 4 & 0 & -2 & h & 0 & \cdots & & & & & \cdots & 0 \\
 & & & & & & & \ddots & & & & & & \\
0 & \cdots & & & & & & \cdots & 0 & -1 & -\frac{h}{3} & -\frac{4}{3} & 0 & 1 & -\frac{h}{3} \\
0 & \cdots & & & & & & \cdots & 0 & -1 & -h & 2 & 0 & -1 & h \\
1 & -\frac{h}{3} & 0 & \cdots & & & & \cdots & 0 & 0 & 0 & -1 & -\frac{h}{3} & -\frac{4h}{3} & 0 \\
-2 & h & 0 & \cdots & & & & \cdots & 0 & 0 & 0 & -2 & -h & 4 & 0
\end{pmatrix}$$

This matrix contains two rows for each point, which expresses the condition that the second and third derivatives are continuous at this point. With this, q_j is an inner point

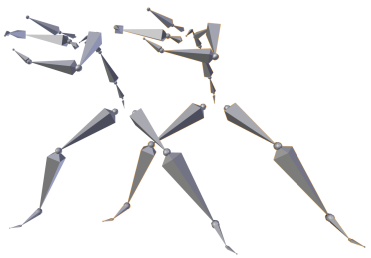


Figure 13: A snapshot from the animation which we approximated using CCBCs through few data points (left: input motion, right: approximated motion).

© 2015 The Author(s)
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd.

iff $(C(q,q'))_{2j-2} = 0$ and $(C(q,q'))_{2j-1} = 0$, where (q,q') is the vector $(q_1,q'_1,q_2,q'_2,...,q_N,q'_N)$ (or, if $d > 1$, a matrix where each column is a vector).

References

[AA00] ANDERSEN E. D., ANDERSEN K. D.: The MOSEK interior point optimizer for linear programming: an implementation of the homogeneous algorithm. In *High performance optimization*. Springer, 2000, pp. 197–232. 6

[BBR11] BAHMANI S., BOUFOUNOS P., RAJ B.: Greedy sparsity-constrained optimization. In *Forty Fifth Asilomar Conference on Signals, Systems and Computers* (2011), pp. 1148–1152. 2, 5

[BD10] BLUMENSATH T., DAVIES M. E.: Normalized iterative hard thresholding: Guaranteed stability and performance. *IEEE J. Sel. Topics Signal Process. 4*, 2 (2010), 298–309. 2, 5

[BE13] BECK A., ELDAR Y. C.: Sparsity constrained nonlinear optimization: Optimality conditions and algorithms. *SIAM J. Optim. 23*, 3 (2013), 1480–1509. 2, 5

[BM98] BERGEAUD F., MALLAT S.: Matching pursuit of images. *Wavelet Analysis and Its Applications 7* (1998), 285–300. 2, 5

[Bur74] BURCHARD H. G.: Splines (with optimal knots) are better. *Applicable Analysis 3*, 4 (1974), 309–319. 2

[CJPT13] CHOUZENOUX E., JEZIERSKA A., PESQUET J.-C., TALBOT H.: A majorize-minimize subspace approach for $\ell_2 - \ell_0$ image regularization. *SIAM Journal on Imaging Science 6* (2013), 563–591. 2, 5

[dB73] DE BOOR C.: Good approximation by splines with variable knots. In *Spline Functions and Approximation Theory*. Springer, 1973, pp. 57–72. 2

[DL14] DENG C., LIN H.: Progressive and iterative approximation for least squares B-spline curve and surface fitting. *Computer-Aided Design 47* (2014), 32–44. 2, 8

[Fou11] FOUCART S.: Hard thresholding pursuit: An algorithm for compressive sensing. *SIAM J. Numer. Anal. 49*, 6 (2011), 2543–2563. 2, 5

[GB04] GRIBOV A., BODANSKY E.: A new method of polyline approximation. In *Structural, Syntactic, and Statistical Pattern Recognition*, vol. 3138 of *Lecture Notes in Computer Science*. Springer, 2004, pp. 504–511. 2

[GI11] GÁLVEZ A., IGLESIAS A.: Efficient particle swarm optimization approach for data fitting with free knot B-splines. *Computer-Aided Design 43*, 12 (2011), 1683–1692. 2, 8

[GP76] GAFFNEY P., POWELL M.: Optimal interpolation. In *Numerical Analysis*, Watson G., (Ed.), vol. 506 of *Lecture Notes in Mathematics*. Springer Berlin Heidelberg, 1976, pp. 90–99. 8

[HP04] HOFER M., POTTMANN H.: Energy-minimizing splines in manifolds. *ACM Trans. Graph. 23*, 3 (2004), 284–293. 4

[HS13] HE L., SCHAEFER S.: Mesh denoising via L_0 minimization. *ACM Trans. Graph. 32*, 4 (2013), 64:1–64:8. 2, 5

[HSvTP12] HILDEBRANDT K., SCHULZ C., VON TYCOWICZ C., POLTHIER K.: Interactive spacetime control of deformable objects. *ACM Trans. Graph. 31*, 4 (2012), 71:1–71:8. 4

[Jup78] JUPP D. L. B.: Approximation to data by splines with free knots. *SIAM J. Numer. Anal. 15* (1978), 328–343. 2

[KA08] KASS M., ANDERSON J.: Animating oscillatory motion with overlap: wiggly splines. *ACM Trans. Graph. 27*, 3 (2008), 28:1–28:8. 4

[KCL*15] KANG H., CHEN F., LI Y., DENG J., YANG Z.: Knot calculation for spline fitting via sparse optimization. *Computer-Aided Design 58* (2015), 179–188. 2, 8

[Kol12] KOLESNIKOV A.: Segmentation and multi-model approximation of digital curves. *Pattern Recognition Letters 33*, 9 (2012), 1171–1179. 2

[LM87] LYCHE T., MØRKEN K.: Knot removal for parametric B-spline curves and surfaces. *Comput. Aided Geom. Des. 4*, 3 (Nov. 1987), 217–230. 2

[LM88] LYCHE T., MØRKEN K.: A data-reduction strategy for splines with applications to the approximation of functions and data. *IMA J. Numer. Anal. 8*, 2 (1988), 185–208. 2

[LXZG05] LI W., XU S., ZHAO G., GOH L. P.: Adaptive knot placement in B-spline curve approximation. *Computer-Aided Design 37*, 8 (2005), 791–797. 2

[Mai14] MAIER G.: Optimal arc spline approximation. *Computer Aided Geometric Design 31*, 5 (2014), 211–226. 2

[MJEM02] MANN R., JEPSON A., EL-MARAGHI T.: Trajectory segmentation using dynamic programming. In *16th Intern. Conf. on Pattern Recognition* (2002), vol. 1, pp. 331–334. 2

[MYH99] MORIYAMA M., YOSHIMOTO F., HARADA T.: Automatic knot placement by a genetic algorithm for data fitting with a spline. In *Proc. Shape Model. Intern.* (1999), pp. 162–169. 2

[NNZC08] NIKOLOVA M., NG M. K., ZHANG S., CHING W.-K.: Efficient reconstruction of piecewise constant images using nonsmooth nonconvex minimization. *SIAM J. Imaging Sciences 1*, 1 (2008), 2–25. 2, 5

[PN14] PATRASCU A., NECOARA I.: Iteration complexity analysis of random coordinate descent methods for ℓ_0 regularized convex problems. arXiv:1403.6622. 1, 2, 5, 9

[PV94] PEREZ J.-C., VIDAL E.: Optimum polygonal approximation of digitized curves. *Pattern Recognition Letters 15*, 8 (1994), 743 – 750. 2

[Sch66] SCHWEIKERT D.: An interpolating curve using a spline in tension. *J. Math. Phys. 45* (1966), 312–317. 4

[SR01] SARFRAZ M., RAZA S. A.: Capturing outline of fonts using genetic algorithm and splines. In *Proc. Intern. Conf. on Information Visualisation* (2001), pp. 738–743. 2

[SvTSH14] SCHULZ C., VON TYCOWICZ C., SEIDEL H.-P., HILDEBRANDT K.: Animating deformable objects using sparse spacetime constraints. *ACM Transactions on Graphics 33*, 4 (2014), 109:1–109:10. 4

[Vas96] VASSILEV T. I.: Fair interpolation and approximation of B-splines by energy minimization and points insertion. *Computer-Aided Design 28*, 9 (1996), 753–760. 2

[XLXJ11] XU L., LU C., XU Y., JIA J.: Image smoothing via L_0 gradient minimization. *ACM Trans. Graph. 30*, 6 (2011), 174:1–12. 2, 5, 8

[YHY03] YOSHIMOTO F., HARADA T., YOSHIMOTO Y.: Data fitting with a spline using a real-coded genetic algorithm. *Computer-Aided Design 35*, 8 (2003), 751–760. 2

[YWS04] YANG H., WANG W., SUN J.: Control point adjustment for B-spline curve approximation. *Computer Aided Design 36*, 7 (2004), 639–652. 2

[ZZYL11] ZHAO X., ZHANG C., YANG B., LI P.: Adaptive knot placement using a GMM-based continuous optimization algorithm in B-spline curve approximation. *Computer-Aided Design 43*, 6 (2011), 598–604. 2

© 2015 The Author(s)
Computer Graphics Forum © 2015 The Eurographics Association and John Wiley & Sons Ltd.