N-D C^k B-Spline Scattered Data Approximation

Nicholas J. Tustison and James C. Gee

Penn Image Computing and Science Laboratory
University of Pennsylvania

Abstract. Since the 1970's B-splines have evolved to become the *de facto* standard for curve and surface representation due to many of their salient properties. Conventional least-squares scattered data fitting techniques for B-splines require the inversion of potentially large matrices. This is time-consuming as well as susceptible to ill-conditioning which leads to undesired results. Lee *et al.* proposed a novel B-spline algorithm for fitting a 2-D cubic B-spline surface to scattered data in [1]. The proposed algorithm utilizes an optional multilevel approach for better fitting results. We generalize this technique to support N-dimensional data fitting as well as arbitrary degree of B-spline. In addition, we generalize the B-spline kernel function class to accommodate this new image filter.

1 Introduction

Given a set of uniformly or nonuniformly distributed data samples, the process of scattered data approximation constructs a smooth surface which best approximates the scattered data. This provides a smooth representation of the scattered data for further analysis. It also allows for the inference of function values at points which are not part of the original scattered data set.

Much research has been done since Riesenfeld's dissertation first introduced B-splines to approximation problems in computer-aided design [2]. The ubiquity of B-splines in graphics, CAD, modeling, etc. is due to many of their salient properties which are discussed in many sources including [3]. The most popular technique for approximation of scattered data using B-splines is least-squares fitting [4]. However, such techniques require the inversion of potentially large matrices which is not only computationally demanding but susceptible to memory problems as well as ill-fitting which leads to undesired results.

Lee *et al.* proposed a B-spline approximation algorithm in [1] which circumvents the problematic issues associated with conventional least-squares fitting for 2-D cubic B-spline surfaces. They also introduce a multilevel approach for better surface fitting. Unfortunately, their algorithm is restricted to 2-D surfaces and cubic B-splines. We present our generalization of Lee's algorithm to include N-D data and arbitrary degree of B-spline. We also describe the necessary modifications to the original B-spline kernel function class to accommodate this generalization.

2 B-Spline Kernel Function

A B-spline is piecewise polynomial function which, in the calculation of B-spline objects, *e.g.* curves, surfaces, etc., acts as a weighting kernel over the set of the control

points. For example, a univariate B-spline curve is defined by the equation

$$C(u) = \sum_{i=0}^{n} P_i N_{i,k}(u) \tag{1}$$

where P_i is the i^{th} control point and $N_{i,k}$ is the i^{th} B-spline of degree k. As can be seen from the above formulation, the B-spline curve is derived as a weighted contribution of the set of control points.

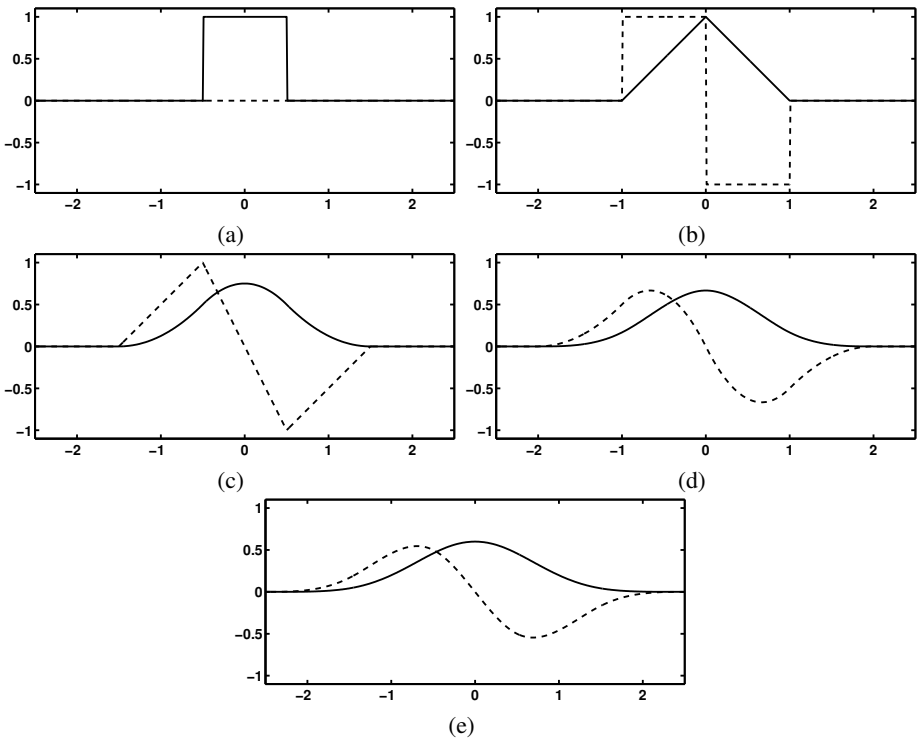

Fig. 1. B-splines of different degrees (solid line) plotted with their derivatives (dashed line). Shown are B-splines of (a) degree 0, (b) degree 1, (c) degree 2, (d) degree 3, and (e) degree 4. Note that the previous incarnation of the B-spline kernel function class is incapable of producing B-splines of degree > 3.

Currently, the B-spline kernel function is only capable of producing B-splines of degree less than or equal to three. However, to generalize to any degree we implemented the Cox-DeBoor recursion relation [3] within the `BSplineKernelFunction` class to generate the necessary polynomial pieces (using the `vnl_real_polynomial` class of the vnl library). This allows one to change the degree of the B-spline kernel function during the existence of a particular instantiation even though it is templated

over a specific order. The templated aspect was kept to maintain backwards compatibility. In addition, we added the following functions:

- `double EvaluateDerivative( const double & u )`
- `MatrixType GetShapeFunctions()`
- `MatrixType GetShapeFunctionsInZeroToOneInterval()`
- `void SetSplineOrder( unsigned int order )`

The first function evaluates the derivative *exactly* at the given parameter value (unlike the derivative calculated in the `BSplineKernelFunctionDerivative` class). This function would render the `BSplineKernelFunctionDerivative` class obsolete. The second function returns a matrix where each row contains the coefficients of a single polynomial piece which, together with the coefficients in the other rows, form the basis function. The third function is similar except it returns the piecewise polynomials in the zero-to-one interval. Finally, the fourth function simply resets the Kernel to be of an arbitrary order. Shown in Figure 1 are both the B-splines of different degrees and their derivatives reproduced from the revised kernel class.

3 B-Spline Scattered Data Approximation

Technical details of the 2-D algorithm (along with pseudocode) are found in Section 3 of [1]. Our generalized algorithm is implemented as an image-to-image filter. The protocol for instantiation of the filter as well as specification of the user-defined parameters is as follows:

```
typedef itk::BSplineScatteredDataImageFilter
        <InputImageType, OutputImageType>  FilterType;
FilterType::Pointer filter = FilterType::New();

filter->SetSplineOrder(3);
filter->SetNumberOfControlPoints(ncps);
filter->SetNumberOfLevels(5);
filter->SetExcludeBackground(true);
filter->SetBackgroundValue(0);
```

The filter is defined by the B-spline order (= polynomial degree), the number of control points in each dimension (which is of type `itk::Array<unsigned int>`), the number of levels ($\geq$ 1), and whether or not to exclude irrelevant pixels which are defined by the background value. Each of these parameters have defaults if they are not set explicitly. If the multilevel approach is used by specifying the number of levels to be greater than one, the number of control points that is specified by the user is the number of control points at the coarsest level of the multilevel scheme.

We illustrate our implementation in Figure 2. Figure 2(a) shows a segmented brain 2-D image slice in which only the pixels associated with gray matter are nonzero. The values of these pixels are used to estimate a fourth order B-spline surface using the multilevel approach.

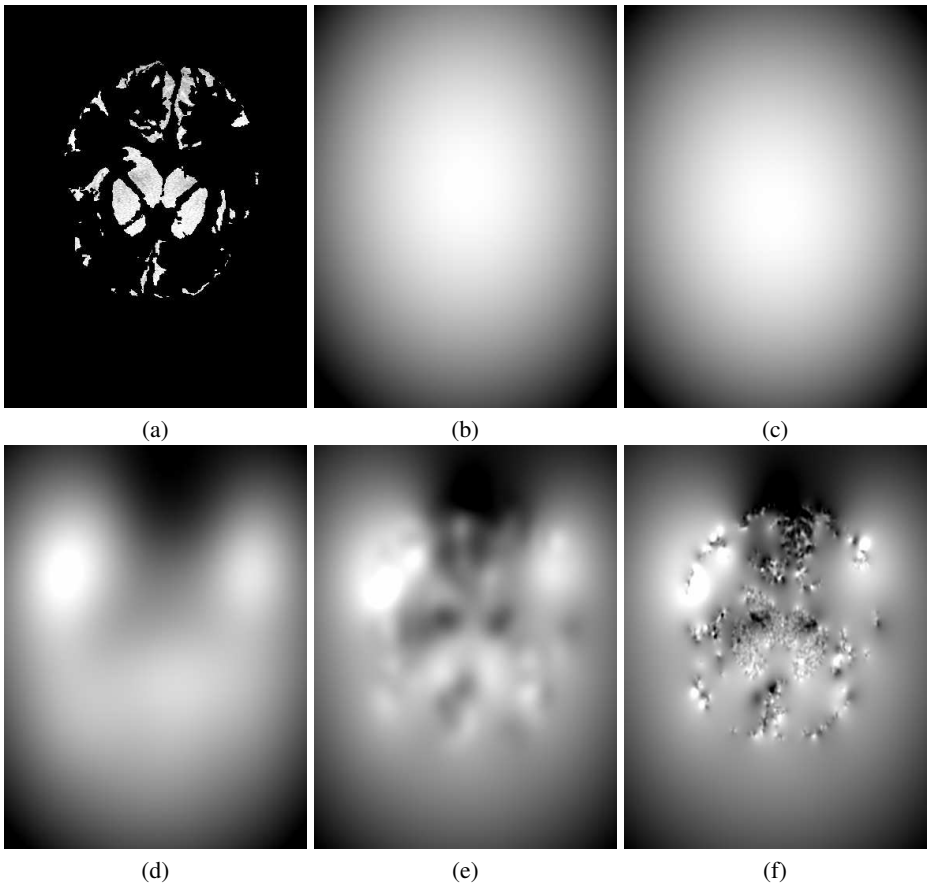

Fig. 2. (a) Original segmented 2-D brain slice showing pixels labeled as gray matter. Using this image we calculate 2-D fourth order B-spline surfaces using a multilevel approach where the coarsest level is defined by 5×5 control points. The number of levels in each of the subsequent images are (b) 1 level, (c) 3 levels, (d) 5 levels, (e) 7 levels, and (f) 10 levels.

References

1. Seungyong Lee, George Wolberg, and Sung Yong Shin, "Scattered data interpolation with multilevel b-splines," *IEEE Transactions on Visualization and Computer Graphics*, vol. 3, no. 3, pp. 228–244, 1997.
2. R. F. Riesenfeld, *Applications of B-Spline Approximation to Geometric Problems of Computer-Aided Design*, Ph.D. thesis, Syracuse University, 1975.
3. L. Piegl and W. Tiller, *The NURBS Book*, Springer-Verlag, New York, NY, 1997.
4. L. Piegl, "On NURBS: A survey," *IEEE Computer Graphics and Applications*, vol. 10, no. 1, pp. 55–71, 1991.

N-D C^k B-Spline Scattered Data Approximation

Nicholas J. Tustison and James C. Gee

Penn Image Computing and Science Laboratory
University of Pennsylvania

Abstract. Since the 1970's B-splines have evolved to become the *de facto* standard for curve and surface representation due to many of their salient properties. Conventional least-squares scattered data fitting techniques for B-splines require the inversion of potentially large matrices. This is time-consuming as well as susceptible to ill-conditioning which leads to undesired results. Lee *et al.* proposed a novel B-spline algorithm for fitting a 2-D cubic B-spline surface to scattered data in [1]. The proposed algorithm utilizes an optional multilevel approach for better fitting results. We generalize this technique to support N-dimensional data fitting as well as arbitrary degree of B-spline. In addition, we generalize the B-spline kernel function class to accommodate this new image filter.

1 Introduction

Given a set of uniformly or nonuniformly distributed data samples, the process of scattered data approximation constructs a smooth surface which best approximates the scattered data. This provides a smooth representation of the scattered data for further analysis. It also allows for the inference of function values at points which are not part of the original scattered data set.

Much research has been done since Riesenfeld's dissertation first introduced B-splines to approximation problems in computer-aided design [2]. The ubiquity of B-splines in graphics, CAD, modeling, etc. is due to many of their salient properties which are discussed in many sources including [3]. The most popular technique for approximation of scattered data using B-splines is least-squares fitting [4]. However, such techniques require the inversion of potentially large matrices which is not only computationally demanding but susceptible to memory problems as well as ill-fitting which leads to undesired results.

Lee *et al.* proposed a B-spline approximation algorithm in [1] which circumvents the problematic issues associated with conventional least-squares fitting for 2-D cubic B-spline surfaces. They also introduce a multilevel approach for better surface fitting. Unfortunately, their algorithm is restricted to 2-D surfaces and cubic B-splines. We present our generalization of Lee's algorithm to include N-D data and arbitrary degree of B-spline. We also describe the necessary modifications to the original B-spline kernel function class to accommodate this generalization.

2 B-Spline Kernel Function

A B-spline is piecewise polynomial function which, in the calculation of B-spline objects, *e.g.* curves, surfaces, etc., acts as a weighting kernel over the set of the control

points. For example, a univariate B-spline curve is defined by the equation

$$C(u) = \sum_{i=0}^{n} P_i N_{i,k}(u) \tag{1}$$

where P_i is the i^{th} control point and $N_{i,k}$ is the i^{th} B-spline of degree k. As can be seen from the above formulation, the B-spline curve is derived as a weighted contribution of the set of control points.

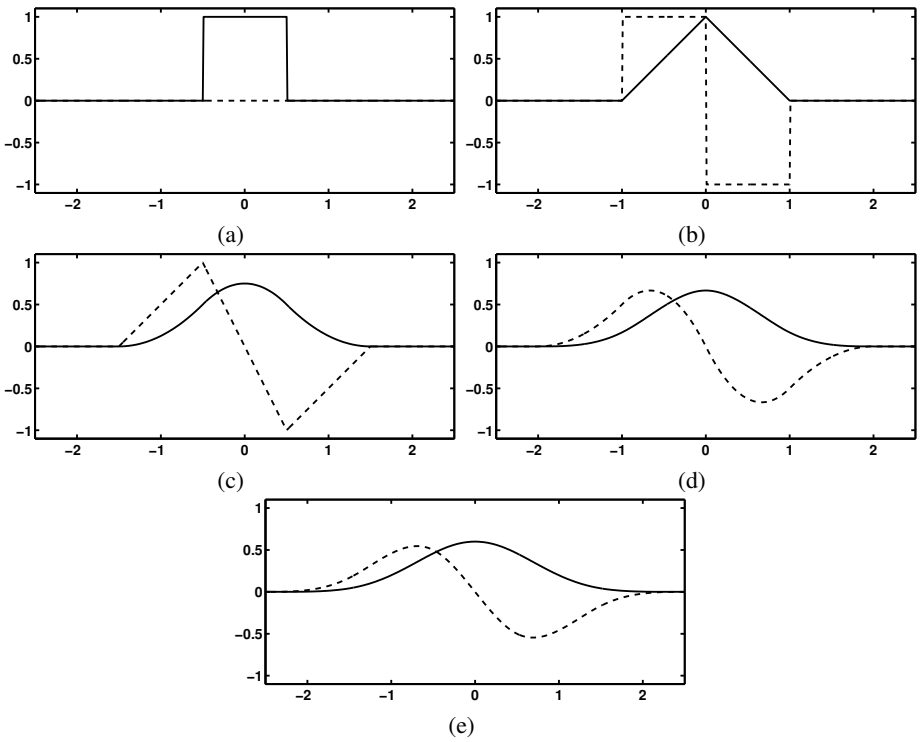

Fig. 1. B-splines of different degrees (solid line) plotted with their derivatives (dashed line). Shown are B-splines of (a) degree 0, (b) degree 1, (c) degree 2, (d) degree 3, and (e) degree 4. Note that the previous incarnation of the B-spline kernel function class is incapable of producing B-splines of degree > 3.

Currently, the B-spline kernel function is only capable of producing B-splines of degree less than or equal to three. However, to generalize to any degree we implemented the Cox-DeBoor recursion relation [3] within the `BSplineKernelFunction` class to generate the necessary polynomial pieces (using the `vnl_real_polynomial` class of the vnl library). This allows one to change the degree of the B-spline kernel function during the existence of a particular instantiation even though it is templated

over a specific order. The templated aspect was kept to maintain backwards compatibility. In addition, we added the following functions:

- `double EvaluateDerivative( const double & u )`
- `MatrixType GetShapeFunctions()`
- `MatrixType GetShapeFunctionsInZeroToOneInterval()`
- `void SetSplineOrder( unsigned int order )`

The first function evaluates the derivative at the given parameter value. This function would render the `BSplineKernelFunctionDerivative` class obsolete. The second function returns a matrix where each row contains the coefficients of a single polynomial piece which, together with the coefficients in the other rows, form the basis function. The third function is similar except it returns the piecewise polynomials in the zero-to-one interval. Finally, the fourth function simply resets the Kernel to be of an arbitrary order. Shown in Figure 1 are both the B-splines of different degrees and their derivatives reproduced from the revised kernel class.

3 B-Spline Scattered Data Approximation

Technical details of the 2-D algorithm (along with pseudocode) are found in Section 3 of [1]. Our generalized algorithm is implemented as an image-to-image filter. The protocol for instantiation of the filter as well as specification of the user-defined parameters is as follows:

```
typedef itk::BSplineScatteredDataImageFilter
        <InputImageType, OutputImageType>  FilterType;
FilterType::Pointer filter = FilterType::New();

filter->SetSplineOrder(3);
filter->SetNumberOfControlPoints(ncps);
filter->SetNumberOfLevels(5);
filter->SetExcludeBackground(true);
filter->SetBackgroundValue(0);
```

The filter is defined by the B-spline order (= polynomial degree), the number of control points in each dimension (which is of type `itk::Array<unsigned int>`), the number of levels (≥ 1), and whether or not to exclude irrelevant pixels which are defined by the background value. Each of these parameters have defaults if they are not set explicitly. If the multilevel approach is used by specifying the number of levels to be greater than one, the number of control points that is specified by the user is the number of control points at the coarsest level of the multilevel scheme.

We illustrate our implementation in Figure 2. Figure 2(a) shows a segmented brain 2-D image slice in which only the pixels associated with gray matter are nonzero. The values of these pixels are used to estimate a fourth order B-spline surface using the multilevel approach.

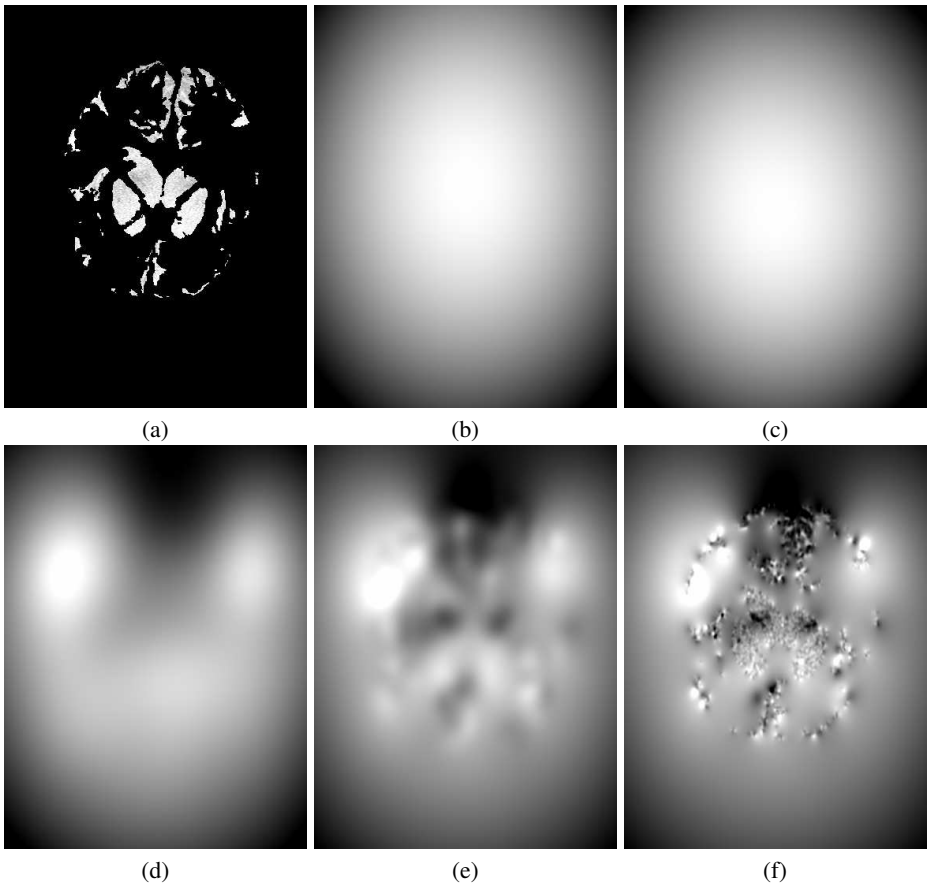

Fig. 2. (a) Original segmented 2-D brain slice showing pixels labeled as gray matter. Using this image we calculate 2-D fourth order B-spline surfaces using a multilevel approach where the coarsest level is defined by 5×5 control points. The number of levels in each of the subsequent images are (b) 1 level, (c) 3 levels, (d) 5 levels, (e) 7 levels, and (f) 10 levels.

References

1. Seungyong Lee, George Wolberg, and Sung Yong Shin, "Scattered data interpolation with multilevel b-splines," *IEEE Transactions on Visualization and Computer Graphics*, vol. 3, no. 3, pp. 228–244, 1997.
2. R. F. Riesenfeld, *Applications of B-Spline Approximation to Geometric Problems of Computer-Aided Design*, Ph.D. thesis, Syracuse University, 1975.
3. L. Piegl and W. Tiller, *The NURBS Book*, Springer-Verlag, New York, NY, 1997.
4. L. Piegl, "On NURBS: A survey," *IEEE Computer Graphics and Applications*, vol. 10, no. 1, pp. 55–71, 1991.