

# Color Image Processing With B-Spline Modeling

Ruşen Öktem<sup>1</sup>, Moncef Gabbouj<sup>2</sup>

Signal Processing Laboratory, Tampere University of Technology

P.O. Box 553, FIN-33101 Tampere, Finland

<sup>1</sup>email: rusen@cs.tut.fi <sup>2</sup> email: moncef@cs.tut.fi

*Abstract*— **B-Splines** have been popular for several purposes. In this work, **B-Splines** are used for modeling noisy multivariate data. A new algorithm is proposed to approximate image data with **B-Spline** surfaces, where image data is assumed to be corrupted by Gaussian noise mixed with outliers. The resulting surface gives a restored continuous representation of discrete data while preserving edges. A gradient operator is applied on the resulting surface to detect edges.

## I. INTRODUCTION

A natural approach in color image processing in the past has been processing each channel separately, such as using monochrome image processing techniques in each color component, which does not utilize the multivariate nature of color images. Recently, vector median has been proposed [1] to utilize the correlation between signal components.

Many applications may need the discrete image data to be represented as a continuous surface. By using some parametric models, a continuous function, either interpolating the data at grid points or passing near them, can be found. In the presence of noise, the latter case is of interest.

Spline functions have been successfully used in modeling scalar data in [3] and multivariate data in [4], when the data is assumed to be corrupted with Gaussian noise. However, in many cases the presence of outliers may deviate the Gaussian noise model. Robust methods can be introduced in the smoothing algorithm to deal with outliers [5].

In this work, cubic tensor product **B-Spline** curves are used to provide a "noise-free" continuous representation of observed image data. In section II, Spline functions and Spline curves are introduced and Spline smoothing is described. Our proposed algorithm for Spline fitting is presented in Section III. Edge detection in color images is presented in Section IV. Simulation results are presented in section V, and conclusions are included in Section VI.

## II. SMOOTHING WITH B-SPLINE SURFACES

### A. B-Splines

Spline functions are piecewise polynomials with derivative continuities of up to a certain order at the knots. Knots are the break points between ad-

acent segments of the polynomials. Since the entire domain is divided into segments, spline functions do not suffer from oscillations.

*Definition II.1:* ([6]) Let  $\mathbf{t} = (t_0, t_1, \dots, t_{g+k})$  be a nondecreasing knot sequence. The  $i$ 'th normalized B-spline of order  $k + 1$  for the knot sequence  $\mathbf{t}$  is denoted as  $B_{i,k+1,t}$  and is defined by the recurrence relation

$$B_{i,k+1,t}(x) = \frac{t_{i+k+1} - x}{t_{i+k+1} - t_{i+1}} B_{i+1,k,t}(x) + \frac{x - t_i}{t_{i+k} - t_i} B_{i,k,t}(x), \quad (1)$$

$$B_{i,1,t}(x) = \begin{cases} 1 & \text{if } x \in [t_i, t_{i+1}], \\ 0 & \text{otherwise.} \end{cases}$$

Let  $S_{k+1,t}$  and  $S_{l+1,s}$  be linear spaces defined on some sets  $X$  and  $Y$  into the real numbers, and spanned by independent **B-Spline** functions,  $B_{i,k+1,t}$  and  $B_{j,l+1,s}$  respectively.  $S_{k+1,t} \otimes S_{l+1,s}$  form a linear space of functions on  $X \times Y$  such that each function  $f(x, y)$  of  $S_{k+1,t} \otimes S_{l+1,s}$  can be written as

$$f(x, y) = \sum_{i=0}^{g-1} \sum_{j=0}^{h-1} \theta_{i,j} B_{i,k+1,t}(x) B_{j,l+1,s}(y), \quad (2)$$

where  $B_{i,k+1,t}$  and  $B_{j,l+1,s}$  are **B-spline** basis functions defined on knot sequences  $\mathbf{t}$  and  $\mathbf{s}$ , respectively.

### B. Smoothing

When modeling data with some parametric functions, two cases are possible: to represent the data exactly by interpolating at grid points, and to use an approximate representation in which the function parameters are determined by minimizing some measure of discrepancy between pixel values and function at grid points. In the presence of noise, it is desired that the function will not interpolate but would pass near the observed data. This case will be the focus of this work.

Assume that we are given a set of sample values  $\mathbf{z} = (\mathbf{z}_{1,1}^T, \mathbf{z}_{1,2}^T, \dots, \mathbf{z}_{1,N_2}^T, \mathbf{z}_{2,1}^T, \dots, \mathbf{z}_{N_1,N_2}^T)^T$ , where  $\mathbf{z}_{i,j}^1$  is the  $3 \times 1$  image data vector corresponding

<sup>1</sup>Throughout this paper, matrices and vectors will be denoted with bold letters.

to the RGB values at pixel  $(x_i, y_j)$ . The smoothing vector spline is the function which minimizes

$$\sum_{i=1}^{N_1} \sum_{j=1}^{N_2} (\mathbf{z}_{i,j} - \mathbf{f}(x_i, y_j))^T \boldsymbol{\Sigma}_{i,j}^{-1} (\mathbf{z}_{i,j} - \mathbf{f}(x_i, y_j)) + \sum_{m=1}^3 \lambda_m J_m(f_m), \quad (3)$$

where

$$J_m(f_m) = \int \int \left( \frac{\partial^2 f_m}{\partial x^2} \right)^2 + 2 \left( \frac{\partial^2 f_m}{\partial x \partial y} \right)^2 + \left( \frac{\partial^2 f_m}{\partial y^2} \right)^2 dx dy, \quad (4)$$

and each component  $f_m(x_i, y_j)$  of  $\mathbf{f}(x_i, y_j)$  are as defined in equation (2).  $\mathbf{z}_{i,j}$  is assumed to be corrupted with Gaussian noise such that

$$\mathbf{z}_{i,j} = \mathbf{f}(x_i, y_j) + \mathbf{v}_{i,j} \quad (5)$$

where  $\mathbf{v}_{i,j} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{i,j})$ .

The parameter  $\lambda_m$  establishes a trade-off between smoothness and goodness of fit, whereas the first term in (3) controls the fit to data and the second term (4) controls the smoothness.

### C. Solving the Spline Function Parameters

Assume that we have chosen appropriate basis functions  $B_{i,k+1,t}$  and  $B_{i,k+1,s}$  in  $x$  and  $y$  directions, respectively. Then, the problem is to find the coefficients  $\hat{\theta}_{i,j}$  of tensor product spline function, which will minimize (3). Therefore, we get the solution<sup>2</sup> by taking the derivative of (3) with respect to  $\hat{\theta}_{i,j}$  and equating it to 0,

$$\hat{\theta} = (\mathbf{P}^T \boldsymbol{\Sigma}^{-1} \mathbf{P} + \mathbf{D}_\lambda \mathbf{S})^{-1} \mathbf{P}^T \boldsymbol{\Sigma}^{-1} \mathbf{z}, \quad (6)$$

where

$$\hat{\mathbf{f}} = \mathbf{P} \hat{\theta} \quad (7)$$

$$\hat{\theta} = (\theta_{0,0,1}, \dots, \theta_{0,0,3}, \dots, \theta_{g-1,h-1,3})^T \quad (8)$$

$$\mathbf{z} = (z_{1,1,1}, \dots, z_{1,1,3}, \dots, z_{N_1, N_2, 3})^T \quad (9)$$

$$\hat{\mathbf{f}} = (f_1(x_1, y_1), \dots, f_3(x_1, y_1), \dots, f_3(x_{N_1}, y_{N_2}))^T \quad (10)$$

$$\lambda = \text{diag}(\lambda_1, \lambda_2, \lambda_3) \quad \mathbf{D}_\lambda = \underbrace{\text{diag}(\lambda, \dots, \lambda)}_{N \text{ times}} \quad (11)$$

$$\boldsymbol{\Sigma}^{-1} = \text{diag}(\boldsymbol{\Sigma}_{1,1}^{-1}, \dots, \boldsymbol{\Sigma}_{N_1, N_2}^{-1}) \quad (12)$$

<sup>2</sup>For simplicity, we choose  $t = s$ ,  $k + 1 = l + 1 = 4$ , and  $N_1 = N_2$ , so that we use the same basis spline functions in  $x$  and  $y$  directions.

$$\mathbf{B}_x = \begin{bmatrix} B_{0,k+1,t}(x_1) & \dots & B_{g-1,k+1,t}(x_1) \\ \vdots & & \vdots \\ B_{0,k+1,t}(x_N) & \vdots & B_{g-1,k+1,t}(x_N) \end{bmatrix} \quad (13)$$

$$\mathbf{P} = \mathbf{B}_x \otimes \mathbf{B}_y \otimes \mathbf{I}_3 \quad (14)$$

$$B_{i,j,x}^l = \int B_{i-k-1}^{(l)}(x) B_{j-k-1}^{(l)}(x) dx,$$

$$\mathbf{S} = (\mathbf{B}_x^2 \otimes \mathbf{B}_y^0 + 2\mathbf{B}_x^1 \otimes \mathbf{B}_y^1 + \mathbf{B}_x^0 \otimes \mathbf{B}_y^2) \otimes \mathbf{I}_3, \quad (15)$$

where  $\mathbf{I}_3$  is the  $3 \times 3$  identity matrix,  $B_{i,k+1,t}$  and  $B_{j,k+1,s}$  are the B-spline basis functions,  $N = N_1 \cdot N_2$ , and  $\otimes$  denotes the Kronecker product.

### D. Choosing the Smoothing Parameter

It can be seen from (3) that  $\lambda_m = 0$  leads to the weighted least squares estimate, whereas  $\lambda_m = \infty$  leads to a perfectly smooth estimate. Hence,  $\lambda_m$  establishes a compromise between the two extreme cases. However, keeping  $\lambda_m$  constant all over the processing image may result in oversmoothing or undersmoothing in some regions. Thompson *et al.* [8] discussed a few methods for choosing  $\lambda$  in the scalar case. One of them uses the sum of residual squares and is based on the estimated noise variance such that

$$\frac{\sum_{i=1}^{N_1} \sum_{j=1}^{N_2} w_{i,j}^2 (z_{i,j} - \hat{f}(x_i, y_j))^2}{N_1 \times N_2} = \sigma^2. \quad (16)$$

For multivariate data, three different ways to estimate  $\lambda_m$  are discussed in [4]. However, in the case of correlated noise, estimating  $\lambda$  as in [4] is computationally expensive. On the other hand, if the covariance matrix is diagonal, each channel can be processed independently. In that case, estimation of  $\lambda$  with (16) can be carried out efficiently with some matrix manipulations [3], [5]. Generally, it may be the case that the covariance matrix is not diagonal, but the data can be transformed to have diagonal covariance matrix.

### E. Diagonalizing the Covariance Matrix

Recall equation (5), and let  $\mathbf{V}$  be the orthonormal matrix where the rows are the normalized eigenvectors of the covariance matrix  $\boldsymbol{\Sigma}$ . If the observation vector is transformed with  $\mathbf{V}$ , one gets

$$\mathbf{Vz} = \mathbf{Vf} + \mathbf{Vv}.$$

Denote  $\mathbf{Vz} = \mathbf{z}_t$  and  $\mathbf{Vv} = \mathbf{v}_t$ . Then  $\mathbf{E}\{\mathbf{v}_t \mathbf{v}_t^T\} = \mathbf{V}\boldsymbol{\Sigma}\mathbf{V}^T$ , where  $\mathbf{V}\boldsymbol{\Sigma}\mathbf{V}^T$  gives a diagonal matrix with eigenvalues of  $\boldsymbol{\Sigma}$  on the diagonal. Consider fitting a spline function to the transformed vector  $\mathbf{z}_t$  such that

$$\hat{\mathbf{f}}_t = \mathbf{P}(\mathbf{P}^T \mathbf{V}\boldsymbol{\Sigma}\mathbf{V}^T \mathbf{P} + \mathbf{D}_\lambda \mathbf{S})^{-1} \mathbf{P}^T \mathbf{V}\boldsymbol{\Sigma}\mathbf{V}^T \mathbf{Vz}.$$

The kronecker product with  $\mathbf{I}_3$  in equations (14,15) ensures that  $\mathbf{VP}^T$ ,  $\mathbf{VD}_\lambda$ ,  $\mathbf{SV}^T$  are commutative, so

$$\hat{\mathbf{f}}_t = \mathbf{VP}(\mathbf{P}^T \mathbf{\Sigma P} + \mathbf{D}_\lambda \mathbf{S})^{-1} \mathbf{P}^T \mathbf{\Sigma z},$$

$$\hat{\mathbf{f}} = \mathbf{V}^T \hat{\mathbf{f}}_t. \quad (17)$$

Equation (17) concludes that transforming the input data with eigenvectors matrix  $\mathbf{V}$  before smoothing does not result in any loss of information in the data.

### III. ALGORITHM

So far, the observation data was assumed to be corrupted with Gaussian distributed noise. However, in many cases, largely deviating observations (outliers) may occur. Therefore, we propose a robust method to deal with outliers, while suppressing Gaussian noise with the smoothing algorithm. Consider the  $n$ 'th step and let  $\tilde{\mathbf{z}}_m^{(n)}$  be the estimate of  $\mathbf{z}_{m_i} = (\mathbf{Vz})_m$  for the  $m$ 'th channel. For  $m = 1, 2, 3$ ,

1. Calculate the residual squares  $r_{m_i}^2 = (z_{m_i} - \tilde{z}_{m_i}^{(n)})^2$ .
2. For  $i = 1, 2, \dots, N$ , replace  $\tilde{z}_{m_i}^{(n)}$  with  $z_{m_i}$ , if  $r_{m_i}^2$  exceeds the variance of that channel.
3. Estimate  $\tilde{\mathbf{z}}_m^{(n+1)} = \mathbf{U}(\mathbf{I} + \lambda_m \sigma_m^2 \mathbf{\Delta})^{-1} \mathbf{U}^T \tilde{\mathbf{z}}_m^{(n)}$  ( $\mathbf{U} \mathbf{\Delta} \mathbf{U} = \mathbf{P}^{-T} \mathbf{S} \mathbf{P}^{-1}$ ,  $\mathbf{U}$  is unitary and  $\mathbf{\Delta}$  is diagonal [3], [5], and  $\sigma_m^2$  is the noise variance of  $m$ 'th channel).
4. Go to step 1, until maximum number of iterations is reached or convergence is achieved.
5. Compute  $\hat{\mathbf{f}} = \mathbf{V}^T \tilde{\mathbf{z}}^{(n_i)}$ .

To start the algorithm,  $\tilde{\mathbf{z}}_m^{(0)}$  is chosen to be the standard median. If the covariance matrix is unknown, the algorithm is applied to each channel separately without transforming the observation vector, and the covariance matrix is estimated between the output of the algorithm and  $\tilde{\mathbf{z}}^{(1)}$ . The final estimate is computed as the spline approximation to  $\tilde{\mathbf{z}}^{(n_i)}$ , where  $\tilde{\mathbf{z}}^{(n_i)}$  is the output of the iterative algorithm.

### IV. EDGE DETECTION

Edge detection has been one of the fundamental tasks in computer vision and image processing. Several methods have been proposed for grayscale and color edge detection. In the presence of noise, a natural choice has been to perform a noise suppression process prior to edge detection. Fitting a surface as described in Section II is a noise suppression process which also offers a continuous representation for the digital image data. Chen and Yang [2] investigated edge detection with spline fitting on grayscale images. We perform a gradient operator on the continuous surface for detecting edges. Di Zenzo [9] extended the gradient based edge detection techniques to multivariate images by applying difference operators on each component and then combining the results by taking the root mean square. We represent the fundamentals of this method in this section.

Let us define  $\mathbf{f} : R^2 \rightarrow R^3$  as a continuous color image. For each processing window, we are interested in the direction through point  $(x, y)$  along which  $f$  has the maximum rate of change and the absolute value of this change. Thus, we want to maximize

$$\nabla_x^2 \mathbf{f} dx^2 + 2\nabla_{xy}^2 \mathbf{f} dx dy + \nabla_y^2 \mathbf{f} dy^2 \quad (18)$$

under the condition  $dx^2 + dy^2 = 1$ , where

$$\nabla_x^2 \mathbf{f} = \left| \frac{\partial f_1}{\partial x} \right|^2 + \left| \frac{\partial f_2}{\partial x} \right|^2 + \left| \frac{\partial f_3}{\partial x} \right|^2,$$

$$\nabla_{xy}^2 \mathbf{f} = \frac{\partial^2 f_1}{\partial x \partial y} + \frac{\partial^2 f_2}{\partial x \partial y} + \frac{\partial^2 f_3}{\partial x \partial y}.$$

Equation (18) can be reformulated as finding  $\theta$  which maximizes

$$\nabla_x^2 \mathbf{f} \cos^2 \theta + 2\nabla_{xy}^2 \mathbf{f} \cos \theta \sin \theta + \nabla_y^2 \mathbf{f} \sin^2 \theta. \quad (19)$$

Taking the derivative of (19) with respect to  $\theta$  and setting it to 0, one gets

$$\theta = \frac{1}{2} \arctan \frac{2\nabla_{xy}^2 \mathbf{f}}{(\nabla_x^2 \mathbf{f} - \nabla_y^2 \mathbf{f})}. \quad (20)$$

After computing  $\theta$ , we substitute it into (19), threshold the resulting value, and mark the present pixel as an edge if it exceeds a specified threshold. We further process to connect missed edges by combining broken contours, and to remove false edges by removing isolated pixels, which were marked as edges.

### V. SIMULATION RESULTS

In the simulations, we used two  $256 \times 256$  images (Figure 1) for testing the noise suppression algorithm. As a measure of fidelity we used PSNR which is defined as

$$10 \log_{10} \frac{255^2}{\frac{1}{N^2} \sum_{i,j=1}^N (\hat{\mathbf{f}}(i,j) - \mathbf{f}_{i,j})^T (\hat{\mathbf{f}}(i,j) - \mathbf{f}_{i,j})}.$$

Noisy Peppers image is obtained by adding 3% outliers and Gaussian noise with  $\sigma^2 = 150$  per channel, where covariance between R-G = 140, R-B = G-B = 0.00. Noisy Lenna image is obtained by adding 2.2% outliers per channel and Gaussian noise, where  $\sigma^2 = 300$  for R and G components,  $\sigma^2 = 70$  for B component, covariances between R-G = 160, R-B = 50, G-B = 10. Filtered images are displayed in Figures 3-5.

For testing the edge detection algorithm quantitatively, Pratt's FOM as defined in [7] is used. The results are displayed in Figures 6-7.

### VI. CONCLUSIONS

An algorithm for modeling color image data with cubic vector B-Spline surfaces is presented. The proposed algorithm obtains a continuous estimate of a color image data from its noisy samples. The

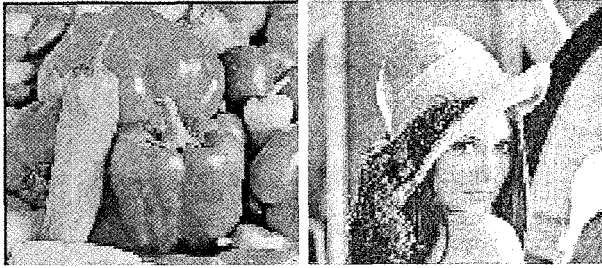


Fig. 1. Original images, *left*: Peppers *right*: Lenna.



Fig. 2. Noisy images, PSNR = *left*: 13.76dB. *right*: 19.83dB.



Fig. 3. Images filtered with  $3 \times 3$  Vector Median filter, PSNR = *left*: 23.75dB, *Right*: 21.51dB.



Fig. 4. Image filtered by Spline fitting with covariance matrix known, PSNR = *left*: 25.23dB, *right*: 24.43dB.



Fig. 5. Image filtered by Spline fitting with covariance matrix unknown, PSNR = *left*: 24.87dB, *right*: 22.49dB.

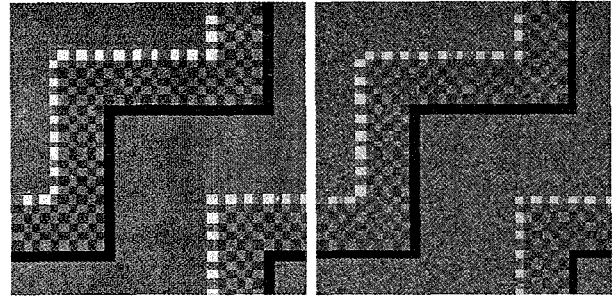


Fig. 6. *left*: Original image. *right*: Noise added image with PSNR = 21.20dB.

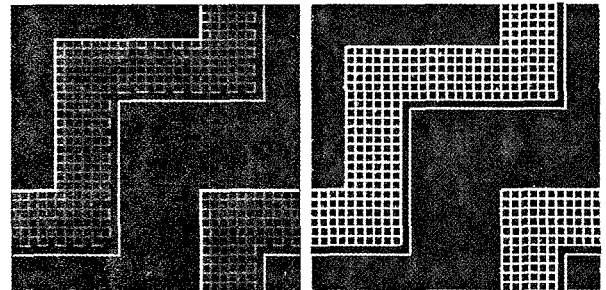


Fig. 7. Edge detection *left*: with Sobel operator, pre-processed with median hybrid filter, postprocessed by thresholding the amplitude, FOM = 0.83, *right* : with spline fitting (covariance matrix known), FOM = 0.97.

estimated surface satisfies a smoothness constraint which is controlled by a smoothing parameter in the algorithm. A gradient operator is applied on the estimated function for extracting edges of a color image. The algorithm is tested in the presence of Gaussian noise mixed with outliers. The results are promising when compared with conventional methods applied on color images.

#### REFERENCES

- [1] J. Astola, P. Haavisto and Y. Neuvo, "Vector Median Filters", *Proceedings of the IEEE*, Vol. 78, No. 4, April 1990.
- [2] G. Chen, Yee H. Hong Yang, "Edge Detection by Regularized Cubic B-Spline Fitting", *IEEE Trans on Systems, Man, and Cybernetics*, Vol. 25, No. 4, pp. 636-643, April 1995.
- [3] P. Craven and G. Wahba, "Smoothing Noisy Data With Spline Functions", *Numerische Mathematik*, **31**, pp. 377-403, 1979.
- [4] J. A. Fessler, "Nonparametric Fixed-Interval Smoothing With Vector Splines", *IEEE Trans. on Signal Processing*, Vol. 39, No. 4, pp. 852-858, April 1991.
- [5] M. Karczewicz and M. Gabbouj, "Robust B-Spline Image Smoothing", *ICIP-94*, Vol. II, 13-16 November, 1994, Austin, Texas.
- [6] C. De Boor, "A Practical Guide to Splines", *Applied Mathematical Sciences*, Vol. 27, Springer-Verlag, 1978.
- [7] W. K. Pratt, *Digital Image Processing*, Wiley Interscience, 1978.
- [8] A. M. Thompson, J. C. Brown, J. W. Kay and D. M. Titterton, "A Study of Methods of Choosing the Smoothing Parameter in Image Restoration by Regularization", *IEEE Trans on Pattern Analysis and Machine Intelligence*, Vol. 13, No. 4, pp. 326-339, April 1991.
- [9] S. Di Zenzo, "A Note on the Gradient of a Multi-Image", *Computer Vision, Graphics, and Image Processing*, **33**, pp. 116-125, 1986.