Math. Commun. **19**(2014), 517–529

Tension splines with application on image resampling

Tina Bosner[1,*], Bojan Crnković[2] and Jerko Škifić[3]

[1] *Department of Applied Mathematics, Faculty of Science, University of Zagreb, Bijenička 30, HR–10 000 Zagreb, Croatia*
[2] *Department of Mathematics, University of Rijeka, Radmile Matejčić 2, HR–51 000 Rijeka, Croatia*
[3] *Department for Fluid Mechanics and Computational Engineering, Faculty of Engineering, University of Rijeka, Vukovarska 58, HR–51 000 Rijeka, Croatia*

Received October 31, 2013; accepted March 14, 2014

Abstract. Digital raster images often need to be represented in higher and lower resolutions. Resampling of digital images is an essential part of image processing. Most efficient and sufficiently accurate image resampling techniques can produce spurious oscillations near sharp transitions of color. To improve that, we introduce tension splines applied dimension by dimension.
The presented tension spline procedure provides an elegant solution to the image resampling by constructing a smooth approximation with sharp non-oscillatory resolution of discontinuities. The numerical results on real digital images are given to show effectiveness of the proposed algorithm.

AMS subject classifications: 65M08, 65D18, 68U10

Key words: image zooming, image resampling, tension spline, interpolation, histospline

1. Introduction

Resampling is the process of interpolating the pixel values based on some original raster image. This is used when the input and output image size do not line up exactly. Efficient methods for resampling of the raster digital images are very important in many applications, ranging from medical imaging, gaming, to electronic publishing. The algorithms need to be very efficient and sufficiently accurate, with a minimum of numerical artefacts in the solution.

A common approach to image resampling, is to turn a discrete RGB image into a continuous function and finally after geometric transformations back to the appropriate discrete image. The most commonly used methods are the nearest neighbour, the bilinear and bicubic interpolation [7, 8]. The above mentioned methods use the polynomial spline interpolation to preserve the image information as much as possible. The bicubic method is the most accurate and visually pleasing of the mentioned methods, but it can produce spurious oscillations and reduced visual sharpness near sharp color transitions.

*Corresponding author. *Email addresses:* tinab@math.hr (T. Bosner), bojan.crnkovic@uniri.hr (B. Crnković), jerko.skific@riteh.hr (J. Škifić)

In the recent years, the methods primarily used for solving PDEs have been adapted to problems in image processing and image resampling [4]. These methods are computationally more demanding, but produce better results when compared to commonly used methods.

We propose a new method for image resampling that is motivated by the image capturing process of digital cameras and a reconstruction approach used in finite volume methods that are primarily developed for solving PDEs and boundary value problems. The new method is based on the reconstruction of surfaces using tension histosplines. The basic idea of the proposed new method is to identify pixels with 2D numerical cells (instead of nodes) and pixel values with cell averages. This method then uses the co-monotone and co-convex tension spline to reconstruct a smooth surface based on the cell averaged values of the pixels. The tension spline will preserve sharpness of the edges in the reconstructed surface that approximates the image. The spline must satisfy the co-convex and co-monoton condition to eliminate or reduce oscillations, that would otherwise be present. After the reconstruction, the image can be transformed to an appropriate discrete form. While some of the methods are specialized for zooming of images by doubling the image size [1, 14], the proposed method can be applied equally well on image upscaling and downscaling and does not depend on the size of the original image.

The paper is organized as follows: in Section 2, the tension spline interpolatiom and its numerical realization are briefly explained. In Section 3, spline interpolation is applied to the 1D reconstruction problem. The basic idea of image resampling is described in Section 4. Section 5 deals with numerical experiments, and it is followed by the conclusion.

2. Co-monotone and co-convex tension spline interpolation

Suppose that point-wise values $v_i = v(x_i)$, $i = 0, \dots, n$ of a function v are given. We are interested in a non–oscillatory spline approximation of the function v. To achieve this, we use tension splines.

Definition 1. *Let $a = x_0 < x_1 < \cdots < x_{n-1} < x_n = b$ be a partition of $[a, b]$, and $p_i \in \mathbb{R}$, $p_i > 0$ for $i = 0, \dots, n-1$, tension parameters. A tension spline is a function s such that*

$$s^{(4)}(x) - p_i^2 s^{(2)}(x) = (D^2 - p_i^2) D^2 s(x) = 0,$$

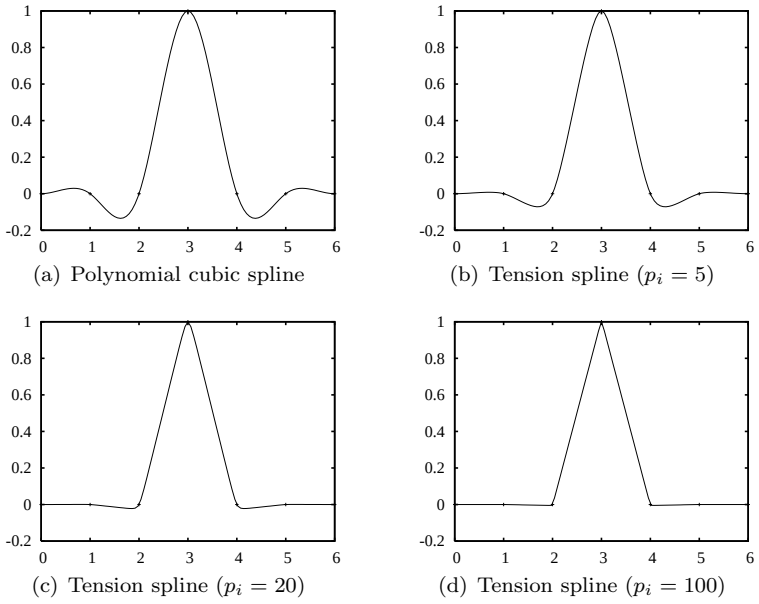
for every $x \in [x_i, x_{i+1})$, $i = 0, \dots, n-1$.

It is easy to check that

$$s|_{[x_i, x_{i+1})} \in \operatorname{span}\{1, x, e^{p_i x}, e^{-p_i x}\} = \operatorname{span}\{1, x, \sinh(p_i x), \cosh(p_i x)\}.$$

The smoothness of the spline at the inner knot x_i, $i = 1, \dots, n-1$, can vary from knot to knot, from being discontinuous up to having continuous second derivative. The most common cases are C^1 [10, 11] and C^2 [10, 12] tension splines.

An interesting property of the tension spline is its behaviour when either all of its tension parameters tend to zero or to infinity. For $p_i = 0$, for all i, the tension spline is equal to a cubic polynomial spline. For $p_i \to \infty$, for all i, the tension spline becomes



(a) Polynomial cubic spline (b) Tension spline ($p_i = 5$)

(c) Tension spline ($p_i = 20$) (d) Tension spline ($p_i = 100$)

Figure 1: *Interpolation by the polynomial cubic spline and uniform tension splines*

a linear polynomial spline (see [6]), thus justifying the name "spline under tension". For all other p_i, a tension spline is "between" a cubic and a linear spline. In the next example, the points $(0,0),(1,0),(2,0),(3,1),(4,0),(5,0),(6,0)$ are interpolated by a polynomial cubic spline and several tension splines with uniform tension parameters (see Figure 1).

An application of tension splines requires a stable way of calculating with them. The Chebyshev theory [2, 22, 23] is an approach that leads to numerically stable algorithms. By that theory, we can define a canonical complete Chebyshev (CCC)–system for tension splines [3]:

$$\mathrm{C}(x) = \begin{cases} \cosh(p_0 x), & \text{for } x \in [x_0, x_1), \\ \dfrac{\cosh(p_0 x_1)}{\cosh(p_1 x_1)} \cdots \dfrac{\cosh(p_{i-1} x_i)}{\cosh(p_i x_i)} \cdot \cosh(p_i x), & \text{for } x \in [x_i, x_{i+1}), \end{cases}$$

which gives a basis that is a generalization of power basis for the polynomials:

$$\begin{aligned} u_1(x) &= 1, \\ u_2(x) &= \int_a^x d\tau_2, \\ u_3(x) &= \int_a^x d\tau_2 \int_a^{\tau_2} \mathrm{C}(\tau_3)\, d\tau_3, \\ u_4(x) &= \int_a^x d\tau_2 \int_a^{\tau_2} \mathrm{C}(\tau_3)\, d\tau_3 \int_a^{\tau_3} \frac{d\tau_4}{\cosh^2(p(\tau_4)\,\tau_4)}. \end{aligned}$$

So, the tension spline is piecewisely spanned by $\{u_1, u_2, u_3, u_4\}$.

Further, the Chebyshev theory suggests that the most convenient basis for a spline space are the B-splines [2, 22, 23], which possess a lot of nice properties, like non–negativity, compact supports, the partition of unity, etc. For calculating with splines as a linear combination of B-splines, we use algorithms based on the knot insertion [2, 3, 19, 21]. Such algorithms are relatively simple and numerically very stable. The algorithms in [3] use some specific functions and their limiting and asymptotic behavior. Therefore, we can safely calculate with tension splines having tension parameters ranging from $p_i = 0$ for some i, up to p_i almost as large as the maximum number the floating point number representation allows (usually not needed). The C^1 tension splines are calculated by the generalized de Boor algorithm, while the C^2 tension splines are calculated from C^1 by an Oslo type algorithm [3]. The generalized derivative formula [3, 18, 20] gives a straightforward way to calculate the first and the second derivatives of a tension spline.

Our interest is the interpolation of function values at the knots by the C^2 tension splines. Beside $n+1$ interpolation conditions for the C^2 tension spline, we need two extra conditions to determine the spline uniquely. We add the first derivative end conditions. For our purpose, we are also interested in a non–oscillatory tension spline approximation of a function, which will determine the tension parameters (see [9]). In this case, the most convenient criterions are:

a) the interpolating tension spline has to be monotone on some subinterval, if the successive interpolation points on that subinterval are monotone;

b) the interpolating tension spline has to be convex (concave), if the successive interpolation points on some subinterval are convex (concave).

In [13], B. J. McCartin suggests algorithms for co-convex interpolation and co-monotone interpolation, which are not based on the B-spline representation. If we define

$$\begin{aligned} m_i &:= \frac{v_{i+1} - v_i}{h_i}, \qquad i = 0, \ldots, n-1 \\ b_0 &:= \frac{v_1 - v_0}{h_0} - f'(a) \\ b_i &:= \frac{v_{i+1} - v_i}{h_i} - \frac{v_i - v_{i-1}}{h_{i-1}}, \qquad i = 1, \ldots, n-1 \\ b_n &:= f'(b) - \frac{v_n - v_{n-1}}{h_{n-1}}, \end{aligned}$$

with $h_i := x_{i+1} - x_i$, these algorithms are based on the theorem [16] which can be simplified as:

(i) if b_i and b_{i+1} are positive (negative), then for p_{i-1}, p_i and p_{i+1} sufficiently large s'' is positive (negative) in $[x_i, x_{i+1}]$;

(ii) for sufficiently large p_{i-1}, p_i and p_{i+1}, s' in $[x_i, x_{i+1}]$ has the same sign as m_i.

Both claims are actually consequences of the fact that a tension spline approaches (locally) a linear spline when (some) tension parameters grow, so if some p_i satisfy (i) or (ii), then the same will be true for any $\bar{p}_i > p_i$.

An inflection point of s is said to be extraneous on $[x_i, x_{i+1}]$ if $b_i b_{i+1} > 0$. If $s''(x_i) \neq 0$, then $s''(x_i)b_i > 0$ is a necessary and sufficient condition for no extraneous inflection points. In the co-convex interpolation, the parameter p_i, $i = 0, \ldots, n-1$, is iteratively altered until a large enough parameter is selected so as to enforce $s''(x_i)b_i > 0$, and therefore avoid extraneous inflection points in the selected subinterval, which is assured by (i) (see [15, 16]).

In the co-monotone interpolation, if the polygonal interpolant has a slope of constant sign in three successive intervals, then the tension parameters in these intervals are selected so that the first derivative of the tension spline has the same sign in the middle interval. The existence of such tension parameters is guaranteed by (ii) (see again [15, 16]). This algorithm also iteratively changes p_i to enforce that the possible extreme point of s' on the interval $[x_i, x_{i+1}]$ has the correct sign.

Our algorithm starts by putting all tension parameters equal to zero, i.e., it starts from the cubic polynomial spline interpolation. Then it alters the tension parameters to fulfill firstly the co-convex, and then the co-monotone conditions. In the algorithms for co-convex and co-monotone interpolations, each iteration starts by solving one strictly diagonally dominant tridiagonal linear system which is well conditioned for every choice of tension parameters and a partition for which the smallest and the largest subinterval do not differ too much [15]. The iterations stop when the maximal relative distance between the new and the old tension parameter becomes less than the given tolerance. The number of iterations is usually relatively small, as in our examples below. For all other details of algorithms see [13].

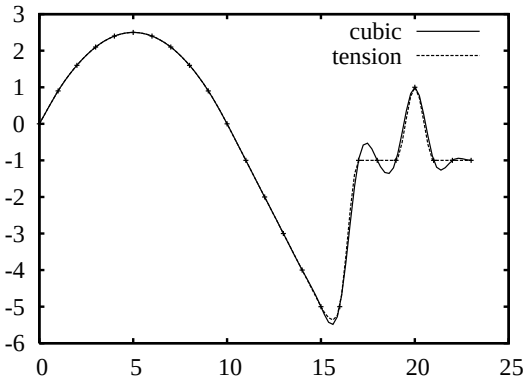


Figure 2: *Cubic spline interpolation and co-convex and co-monotone tension spline interpolation*

Also, both algorithms keep p_i equal to zero or small, if cubic interpolation satisfies or "almost" satisfies the convex or monotone conditions (see Figure 2). The exception is the case when the three successive interpolation points lie (or almost lay) on the straight line. In that case the algorithm puts large tension parameters ($p_i = 1000$) to enforce almost a straight line between these three points. Finally, the tension parameters obtained by previous algorithms are used to get the resulting interpolating tension spline in the B-spline representation.

3. Tension histospline

Consider the following reconstruction problem. Suppose that $a = x_0 < x_1 < \cdots < x_{n-1} < x_n = b$ and that instead of the point-wise values, the cell averages $\overline{v}_i$ of the function v,

$$\overline{v}_i = \frac{1}{\Delta x_i} \int_{x_{i-1}}^{x_i} v(x)\,\mathrm{d}x, \quad i = 1, \ldots, n, \tag{1}$$

with $\Delta x_i := x_i - x_{i-1}$, for all numerical cells $I_i = [x_{i-1}, x_i]$ are known.

We are interested in the high order non–oscillatory approximation s of the function v such that

$$\frac{1}{\Delta x_i} \int_{I_i} s(\xi)\,\mathrm{d}\xi = \overline{v}_i, \tag{2}$$

for all i.

Requirement (2) is called a histopolation problem and the spline function s that satisfies (2) is called a histospline. The solution to this problem is based on the interpolation problem in Section 2. First, the primitive function of the given function v,

$$V(x) = \int_a^x v(\xi)\,\mathrm{d}\xi \tag{3}$$

is defined. Notice that since the cell averages of v are known (1), the values of V at the cell boundaries are explicitly given by

$$V(x_i) = \sum_{j=1}^{i} \int_{x_{j-1}}^{x_j} v(\xi)\,\mathrm{d}\xi = \sum_{j=1}^{i} \overline{v}_j \Delta x_j. \tag{4}$$

Using the previously described spline interpolation procedure for the primitive function V, we can obtain the tension spline S that interpolates the function V at the cell edges x_{i-1} and x_i. Finally, the function

$$s = S' \tag{5}$$

is the required non–oscillatory function that satisfies the condition (2). The co-convex and co-monotone conditions for the interpolating spline S reduce oscillations of the histospline s.

4. Application to image resampling

Although a simplified model, one can say that digital camera sensors "record" photons that are absorbed on non-overlapping, light sensitive areas of the sensor over a relatively short time period. Instead to consider a pixel of an image as a grid point of the rectilinear $m \times n$ grid, each pixel is treated as a 2D numerical cell and the value of the pixel is interpreted as the cell averaged light intensity. No extra treatment of raw image data is necessary. This interpretation leads to the histopolation approach to image resampling. The histopolation approach agrees with the image capturing process of digital cameras.

Similarly to the reconstruction problem from the previous section, the value of the pixel is the cell average $\overline{v}_{i,j}$ of a function v in the numerical cell:

$$\overline{v}_{i,j} = \frac{1}{\|I_{i,j}\|} \int_{I_{i,j}} v(x,y)\,\mathrm{d}x\,\mathrm{d}y. \tag{6}$$

Our goal is to obtain the average values of pixels on some other $\tilde{m} \times \tilde{n}$ grid that shares the same boundaries as the original grid. The unknown function v will be approximated by a tension histospline surface s. The spline surface s must satisfy the reconstruction condition (6) for all cells in the rectilinear grid.

The histospline surface can be obtained as the tensor product of two univariate methods which compute the tension histosplines. In this way, the 2D reconstruction and resampling can be done dimension by dimension. This approach is a computationally rational one and it is a straightforward generalization of the 1D reconstruction problem.

It is enough to consider the problem in 1D and resample the image in one direction. The original grid contains n numerical cells (pixels) in each row, while the resampled grid contains $\tilde{n}$ of them:

$$\tilde{I}_i = [\tilde{x}_{i-1}, \tilde{x}_i],\ i = 1 \ldots \tilde{n},$$

where $\tilde{x}_0 = x_0$ and $\tilde{x}_{\tilde{n}} = x_n$. To resample the image in one direction it is enough to use the reconstructed histospline (5):

$$\tilde{\overline{v}}_i = \frac{1}{\Delta\tilde{x}_i} \int_{\tilde{x}_{i-1}}^{\tilde{x}_i} s(x)\,\mathrm{d}x = \frac{1}{\Delta\tilde{x}_i} \left(S(\tilde{x}_i) - S(\tilde{x}_{i-1})\right), \quad i = 1, \ldots, \tilde{n}. \tag{7}$$

After resampling is done, for all the rows of the original image, the same resampling procedure must be applied in the other direction.

From the right-hand side of equation (7) it is obvious that it is not necessary to explicitly construct the histospline function s. To obtain the average pixel values of the image pixels it is enough to evaluate interpolating tension splines at the new cell boundaries, as introduced in Section 2. The interpolated pixel values need to be quantized to fit the chosen discretization of pixel intensity values. Typically, 256 levels suffice to represent the pixel intensity.

5. Numerical tests

In order to evaluate the tension spline method (7), four different images were resampled with the proposed method and compered with the bilinear, bicubic [7, 8] and the biquadratic histospline [17] methods. It can be noted that the biquadratic histospline is a special case of the tension histospline, with its tension parameters set to zero. The resulting images are then compared with the reference image by using Mean-Squared Error and Peak Signal-to-Noise Ratio [5] measures, commonly used measures of image quality.

Let A be the original $m \times n$ monochrome image and $\widetilde{A}$ its approximation. The mean squared error (MSE) is defined as:

$$MSE = \frac{1}{m\,n} \sum_{i=1}^{m} \sum_{j=1}^{n} \Big(A(i,j) - \widetilde{A}(i,j) \Big)^2 ,$$

where m and n are image width and height, respectively. For color images with three RGB values per pixel the MSE is defined as follows:

$$MSE = \frac{1}{3} \frac{1}{m\,n} \sum_{c=1}^{3} \sum_{i=1}^{m} \sum_{j=1}^{n} \Big(A(i,j,c) - \widetilde{A}(i,j,c) \Big)^2 ,$$

where c denotes the red, green or blue intensity value per pixel.

The peak signal–to–noise ratio ($PSNR$) is defined as:

$$PSNR = 10 \cdot \log_{10} \left(\frac{A_{max}^2}{MSE} \right) ,$$

where A_{max} is the maximum pixel value of the image.

"Better" approximations have their MSE smaller, but their $PSNR$ larger.

In order to compare the tension histospline's computational cost to other methods, CPU times were recorded for all test images by running the programs on the Intel(R) Core(TM) i7-3770 CPU 3.4 GHz processor.

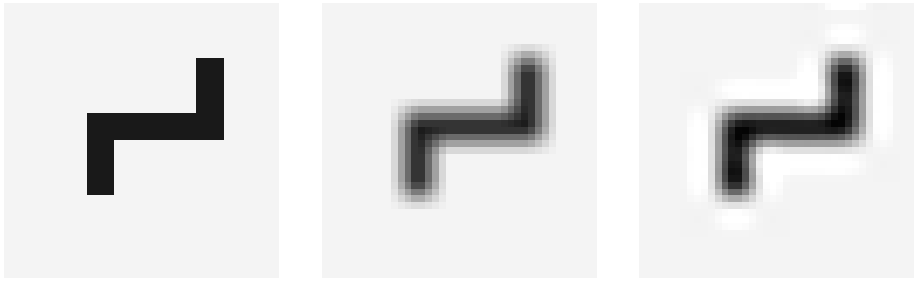
5.1. Resampling a simple greyscale vector image

The vector image shown in Figure 3(a) is created with two different resolutions, i.e., 10×10 and 40×40 pixels. The smaller image is then resampled to higher resolution (Figure 3), while the corresponding MSE and $PSNR$ values are given in Table 5.1.

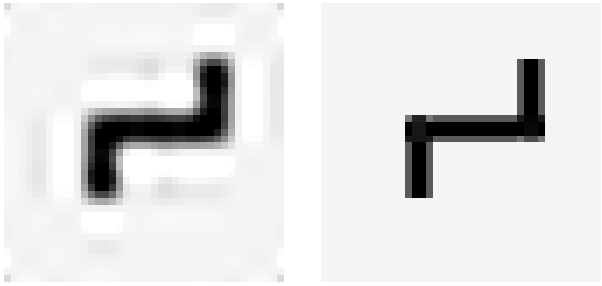
Method	MSE	$PSNR$
tension histospline	199.57	25.13
biquadratic histospline	467.70	21.43
bicubic	579.47	20.50
bilinear	747.22	21.43

Table 1: *Upsampling of the simple greyscale vector image from 10×10 to 40×40*

Numerical results clearly show that the tension histospline, in contrast to other tested methods, is capable of reproducing sharp edges with no apparent noise, at the numerical cost of iterative calculation of co-convex and comonotone parameter p_i, which took on average 6.38 iterations for this example. Measured CPU times for the tension histospline, biquadratic histospline, bicubic and bilinear methods were 4 ms, 0.05 ms, 0.036 ms and 0.02 ms, respectively.



(a) Simple greyscale vector image (b) Bilinear (c) Bicubic



(d) Biquadratic histospline (e) Tension histospline

Figure 3: *Upsampling of an image from 10×10 to 40×40*

5.2. Resampling a smooth grayscale image

To illustrate the fact that the tension histospline behaves like the bicubic method when interpolating smooth data, a simple image with smooth transition of colors, shown in Figure 4, with resolution 874×800 was upsampled with scale factor 2 by the tension histospline and the bicubic method.

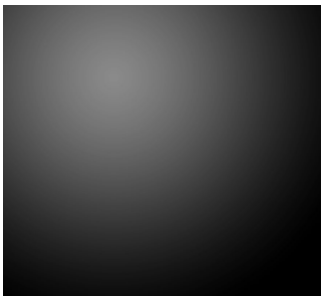


Figure 4: *Grayscale palette*

Both resulting approximations have no visual differences, therefore we put only one image for this example. To demonstrate small numerical differences between these two approximations, we calculated MSE between them, instead of comparing each of the approximations with the original. The small value of the resulting MSE,

which is equal to 0.0017, proves a lack of visual differences between them at the cost of achieving co-monotonicity and co-convexicity in 6.23 iterations on average. The recorded CPU times resulted in 6600 ms for the tension histospline and 3.2 ms for the bicubic method.

5.3. Resampling grayscale vector image

The image shown in Figure 5 is a vectorized image converted to two raster images with resolutions 412×425 and 1648×1702 pixels, respectively. The lower resolution image is then upsampled to match the resolution of the higher resolution. Additionally, the higher resolution image is downsampled to match the resolution of the lower resolution. Both processes were conducted with all four numerical methods.

Figure 5: *Tiger*

Method	MSE	$PSNR$
tension histospline	241.75	24.29
biquadratic histospline	309.95	23.21
bicubic	361.14	22.55
bilinear	424.30	21.85

Table 2: *Upsampling of the tiger image from 412×425 to 1648×1702*

Method	MSE	$PSNR$
tension histospline	26.20	33.94
biquadratic histospline	26.20	33.94
bicubic	134.79	26.83
bilinear	124.36	27.18

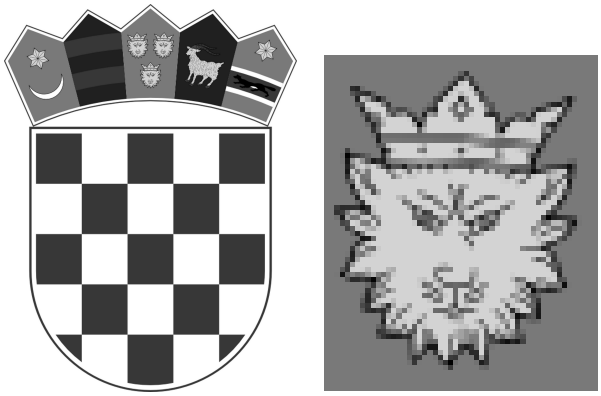
Table 3: *Downsampling of the tiger image from 1648×1702 to 412×425*

The choice of the numerical method exhibits no influence on the visual quality of the resampled image and therefore the resulting images are not shown. However,

a significant influence over the formal metrics is evident, as presented in Tables 2 and 3, where the results of the tension histospline method are better compared to other presented methods. The average number of iterations needed to achieve co-monotonicity and co-convexicity during upsampling was 17.75, while execution times for tension histospline, biquadratic histospline, bicubic and bilinear methods were 8550 ms, 414 ms, 5 ms and 3.1 ms, respectively.

5.4. Resampling a color image

The image shown in Figure 6(a) is a vectorized image converted to two raster images with resolutions 648×859 and 2592×3434 pixels, respectively. The lower resolution image is then upsampled to match the resolution of the higher resolution image.



(a) Coat of arms (b) Zoomed detail

Figure 6: *Color image and a zoomed detail*

Method	MSE	$PSNR$
tension histospline	70.78	29.63
biquadratic histospline	111.73	27.65
bicubic	130.54	26.97
bilinear	152.38	26.30

Table 4: *Upsampling of the coat of arms from 648×859 to 2592×3434*

The zoomed detail of the resampled images obtained by all four methods presented in Figure 7 shows significant differences in the visual quality of resampled images. Tension histospline resampling generates more visually pleasing images, especially around the sharp discontinuities in the image pixel intensities, at the average numerical cost of 12.82 iterations in order to satisfy the co-monotone and co-convex criteria. The obtained CPU times for tension histospline, biquadratic histospline, bicubic and bilinear methods were 59200 ms, 3913 ms, 15 ms and 0.85 ms, respectively. Visual impressions are verified by the results of the formal metrics, as presented in Table 4.

(a) Reference image (b) Bilinear (c) Bicubic

(d) Biquadratic histospline (e) Tension histospline

Figure 7: *Coat of arms zoom detail resampled from 648×859 to 2592×3434*

6. Conclusion

Using the tension histospline to resample an image is a generalization of the quadratic histospline procedure [17]. The tension histospline results on real and test images show that this method gives very good results on image upsampling for arbitrary scale factors. The resampled images have smaller MSE and they also look sharper when compared to commonly used methods. The drawback of this approach is the computational cost of the iterative process in the co-monotone and co-convex interpolation algorithms, and in the calculations of the tension B-splines, which are more complicated than the polynomial ones. Therefore, the tension histospline method has approximately 10 up to 100 times higher execution time compared to the biquadratic histospline method. Nevertheless, the higher computational cost is justified by an obvious higher quality. Reducing the computational cost and re-examining co-monotone and co-convex algorithms could be interesting for some future research.

References

[1] S. Battiato, G. Gallo, F. Stanco, *A locally adaptive zooming algorithm for digital images*, Image and Vision Computing **20**(2002), 805–812.

[2] T. BOSNER, *Knot insertion algorithms for Chebyshev splines*, Ph.D. thesis, Department of Mathematics, University of Zagreb, 2006.
[3] T. BOSNER, M. ROGINA, *Non-uniform exponential tension splines*, Numer. Algorithms **46**(2007), 265–294.
[4] R. GAO, J. SONG, X. TAI, *Image Zooming Algorithm Based on Partial Differential Equations*, Int. J. Numer. Anal. Model. **6**(2009), 284–292.
[5] R. C. GONZALEZ, R. E. WOODS, *Digital Image Processing*, Addison-Wesley, 3rd Edition, New York, 1992.
[6] C. GRANDISON, *Behaviour of exponential splines as tensions increase without bound*, J. Approx. Theory **89**(1997), 289–307.
[7] H. S. HOU, H. C. ANDREWS, *Cubic splines for image interpolation and digital filtering*, IEEE Trans. Acoust., Speech, Signal Processing **26**(1978) 508–517.
[8] R. KEYS, *Cubic convolution interpolation for digital image processing*, IEEE Trans. Acoust., Speech, Signal Processing **29** (1981), 1153–1160.
[9] B. I. KVASOV, *Shape -Preserving Spline Approximation*, World Scientific, Singapore, 2000.
[10] M. MARUŠIĆ, *Stable calculation by splines in tension*, Grazer Math. Ber. **328**(1996), 65–76.
[11] M. MARUŠIĆ, *A fourth/second order accurate collocation method for singularly perturbed two-point boundary value problems using tension splines*, Numer. Math. **88**(2001), 135–158.
[12] M. MARUŠIĆ, M. ROGINA, *A collocation method for singularly perturbed two-point boundary value problems with splines in tension*, Adv. Comput. Math. **6**(1996), 65–76.
[13] B. J. MCCARTIN, *Computation of exponential splines*, SIAM J. Sci. Stat. Comput. **11**(1990), 242–262.
[14] R. M. PIDATELLA, F. STANCO, C. SANTAERA , *ENO/WENO interpolation methods for zooming of digital images*, Image and Vision Computing **20**(2002), 805–812.
[15] S. PRUESS, *Properties of splines in tension*, J. Approx. Theory **17**(1976), 86–96.
[16] S. PRUESS, *Alternatives to the exponential spline in tension*, Math. Comp. **33**(1979), 1273–1281.
[17] N. ROBIDOUX, A. TURCOTTE, M. GONG, A. TOUSIGNANT, *Fast Exact Area Image Upsampling with Natural Biquadratic Histosplines*, in: *Proceedings of the 5th international conference on Image Analysis and Recognition*, (A. Campilho and M. Kamel, Eds.), Springer-Verlag Berlin, Heidelberg, 2008, 85–96.
[18] M. ROGINA, *Basis of splines associated with some singular differential operators*, BIT **32**(1992), 496–505.
[19] M. ROGINA, *On construction of fourth order Chebyshev splines*, Math. Commun. **4**(1999), 83–92.
[20] M. ROGINA, *Algebraic Proof of the B-Spline Derivative Formula*, in: *Proceedings of the Conference on Applied Mathematics and Scientific Computing*, (Z. Drmač, M. Marušić and Z. Tutek, Eds.), 2005, 273–282.
[21] M. ROGINA, T. BOSNER, *On Calculating with Lower Order Chebyshev Splines*, in: *Curves and Surfaces Design*, (P. J. Laurent, P. Sabloniere and L. L. Schumaker, Eds.), Nashville, 2000, 343–353.
[22] L L. SCHUMAKER, *On Tchebycheffian Spline Functions*, J. of Approx. Theory **18**(1976), 278–303.
[23] L. L. SCHUMAKER, *Spline Functions: Basic Theory*, John Wiley & Sons, New York, 1981.