EFFICIENT IMPLEMENTATION OF HIGHER ORDER IMAGE INTERPOLATION

Roland Perko
Graz University of Technology
Inffeldgasse 16
A–8010 Graz
Austria
perko@icg.tu-graz.ac.at

Horst Bischof
University of Technology Graz
Inffeldgasse 16
A–8010 Graz
Austria
bischof@icg.tu-graz.ac.at

ABSTRACT

This work presents a new method of fast cubic and higher order image interpolation. The evaluation of the piecewise n-th order polynomial kernels is accelerated by transforming the polynomials into the interval $[0, 1]$, which has the advantage that some terms of the polynomials disappear, and that several coefficients could be precalculated, which is proven in the paper. The results are exactly the same as using standard n-th order interpolation, but the computational complexity is reduced. Calculating the interpolation weights for the cubic convolution only needs about 60% of the time compared to the classical method optimized by the *Horner's rule*. This allows a new efficient implementation for image interpolation.

Keywords
Interpolation, Reconstruction, Performance

1 INTRODUCTION

Image interpolation is the process of reconstructing a spatially continuous image from a set of discrete equidistant samples. It is fundamental in many image processing operations, such as magnification, subpixel translation, rotation, deformation or warping. These general operations require image values at locations from which no sample is available. From sampling theory it is well known that the sinc-function is the ideal interpolation kernel, which, however, cannot be used in practise [Unser99]. In order to obtain acceptable reconstruction in terms of computational speed and mathematical precision, it is required to design a kernel that is of finite extent and approximates the sinc-function as much as possible.

This is commonly done by symmetrical piecewise n-th order polynomial kernels, where the classical cubic convolution kernel was first introduced to image processing by [Rifma73]. A quantitative evaluation of these interpolation kernels can be found in [Meije99], [Meije01] and [Lehma99].
For arbitrary image warping every pixel of the resulting image has to be interpolated, which is of higher complexity than the evaluation of the geometric warping function itself. Therefore a speed up of the image interpolation task is desired and presented in this paper.

The paper is structured as follows. First, the concept of 1-D signal reconstruction using symmetric piecewise n-th order polynomial kernels is presented (section 2). Next, the transformation of the polynomials is described (section 3). In section 4 an analysis of theoretical and practical performance of the proposed method is given. Finally, concluding remarks are made in section 5.

2 RECONSTRUCTION

As shown in [Meije99] the symmetrical piecewise n-th order polynomial kernel, which approximates the sinc function in $[-(n-1),(n-1)]$, is defined as follows:

$$h(x) = \begin{cases} \sum_{j=0}^{n} a_{ij}|x|^j & \text{if } i \le |x| < i+1 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

To insure continuous, smooth interpolation, it is necessary to set constraints at the knots. Flat field interpolation requires that $\sum_{-\infty}^{\infty} h(x-m) = 1$, where $m = (n+1)/2$. Which leads to two constraints:

$$h(0) = 1 \text{ and } h(x) = 0 \text{ for } |x| = 1, \cdots, m-1 \quad (2)$$

$$h^{(l)}(x) \text{ must be continuous at } |x| = 0, 1, \cdots, m \quad (3)$$

With $l = 0, 1, \cdots, k$ where $k = 0$ for $n = 1$ and $k = n-2$ for $n > 1$ which is proven in [Meije99].

2.1 Cubic kernel example

The cubic 1-D kernel function is build up of third-order polynomials and approximates the sinc-function in the interval $[-2, 2]$. The kernel is given as:

$$h_C(x) = \begin{cases} a_{03}|x|^3 + a_{02}|x|^2 + a_{01}|x| + a_{00} & \text{if } 0 \le |x| < 1 \\ a_{13}|x|^3 + a_{12}|x|^2 + a_{11}|x| + a_{10} & \text{if } 1 \le |x| < 2 \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

With the defined constraints in equation 2 and 3 we get seven equations in eight unknowns. By allowing $a_{13} = \alpha$ to be a tuneable parameter, the system can be solved, yielding following values: $a_{03} = (\alpha+2)$, $a_{02} = -(\alpha+3)$, $a_{01} = 0$, $a_{00} = 1$, $a_{13} = \alpha$, $a_{12} = -5\alpha$, $a_{11} = 8\alpha$ and $a_{10} = -4\alpha$. The free parameter α is chosen that way, that it satisfies the flatness constraint which leads to $\alpha = -\frac{1}{2}$ [Keys81]. The cubic convolution kernel by Keys is given by the following equation and is shown in figure 1.

$$\begin{aligned} p_0(x) &= \frac{3}{2}|x|^3 - \frac{5}{2}|x|^2 + 1 \quad \text{if } 0 \le |x| < 1 \\ p_1(x) &= -\frac{1}{2}|x|^3 + \frac{5}{2}|x|^2 - 4|x| + 2 \quad \text{if } 1 \le |x| < 2 \end{aligned} \quad (5)$$

The standard way to calculate the interpolation weights is to evaluate the piecewise polynomials functions on the supporting points. Every polynomial is evaluated for the according distance ξ. In case of cubic convolution these distances are ξ, $1-\xi$, $\xi+1$ and $2-\xi$ where $\xi \in [0, 1]$.

3 TRANSFORMATION OF THE POLYNOMIALS

To avoid the calculation of these distances all polynomials are transformed in the space $[0, 1]$. This is done by evaluating the polynomial $p_0(x)$ at $\xi = x$ and $\xi = 1-x$, $p_1(x)$ at $\xi = x+1$ and $\xi = 2-x$ analytically for the cubic case. For higher (n-th) order kernels the strategy is the same and we get $n+1$ polynomials, for i odd and $k = i/2$:

$$\begin{aligned} f_i(x) &= p_k(x+k) \\ f_{i+1}(x) &= p_k((k+1)-x) \end{aligned} \quad (6)$$

With

$$f_i(x) = \sum_{j=0}^{n} a_{ij}x^j \quad x \in [0, 1] \quad (7)$$

The cubic convolution kernel is shown in figure 1 and figure 2 shows the transformed cubic polynomials.

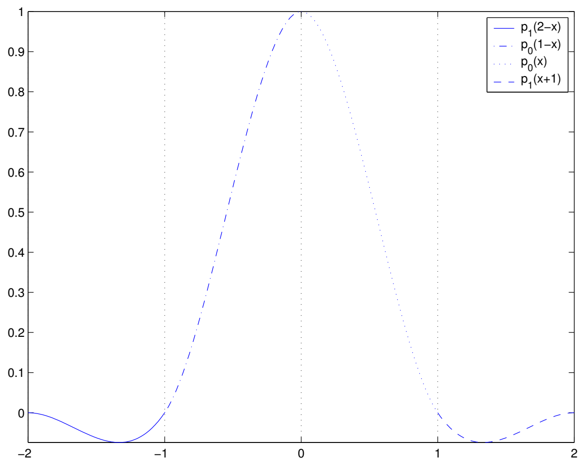


Figure 1: Key's cubic convolution kernel.

In order to save computational time we can proof the following 5 lemmas about the properties of the transformed polynomial coefficients. The proofs are given in the appendix. Figure 3 shows the statements of the lemma graphically on the example of quintic interpolation coefficients.

Lemma 1: The lowest order coefficients a_{i0} are 0 for all polynomials $f_i(x)$, except for the first polynomial $f_0(x)$ where the coefficient is 1.

Lemma 2: The highest order coefficients a_{in} of pairwise symmetrical polynomials, $f_i(x)$ and $f_{i+1}(x)$, i odd, have the same values with opposite sign.

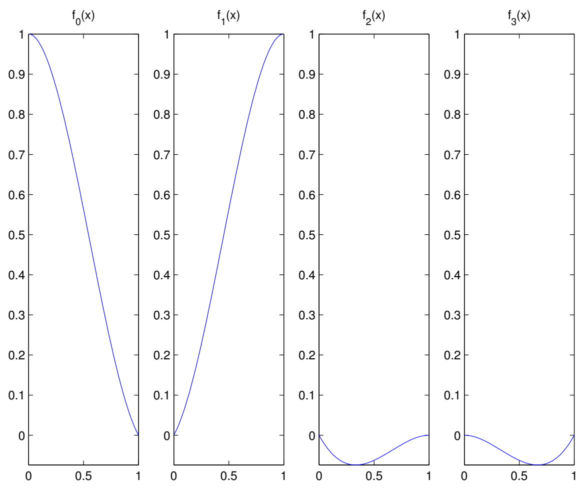
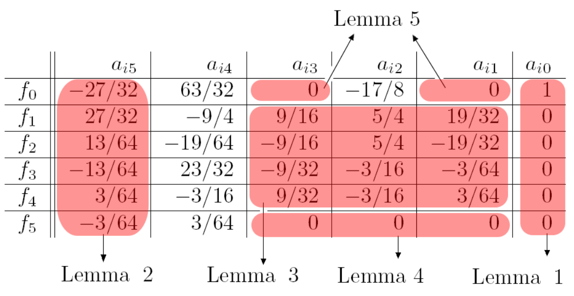


Figure 2: Key's cubic convolution kernel transformed to $[0, 1]$.

Lemma 3: All coefficients of order $j \leq n - 2$ of pairwise non symmetrical polynomials have the same coefficients of odd order and the same coefficients with opposite sign for even order.

Lemma 4: All coefficients a_{nj} with $j \leq n - 2$ of the polynomial $f_n(x)$ are 0.

Lemma 5: All coefficients a_{0j} with odd order j with $j \leq n - 2$ are 0 for the polynomial $f_0(x)$.



Lemma 5
Lemma 2
Lemma 3
Lemma 4
Lemma 1

	a_{i5}	a_{i4}	a_{i3}	a_{i2}	a_{i1}	a_{i0}
f_0	$-27/32$	$63/32$	0	$-17/8$	0	1
f_1	$27/32$	$-9/4$	$9/16$	$5/4$	$19/32$	0
f_2	$13/64$	$-19/64$	$-9/16$	$5/4$	$-19/32$	0
f_3	$-13/64$	$23/32$	$-9/32$	$-3/16$	$-3/64$	0
f_4	$3/64$	$-3/16$	$9/32$	$-3/16$	$3/64$	0
f_5	$-3/64$	$3/64$	0	0	0	0

Figure 3: Statements of the given lemma shown on the example of quintic (5-th order) interpolation coefficients.

Using these 5 lemmas we can formulate following efficient calculation. The distance x has the same value for all polynomials, so the powers of x can be precalculated. Furthermore the multiplications with a power of x and the pairwise same coefficients have to be calculated only once and coefficients with zeros could be skipped.

3.1 Example for the cubic case

The classical cubic convolution kernel according to Keys is given by equation 5.

Transforming the polynomials to the space $[0, 1]$ gives 4 polynomials:

$$\begin{aligned} f_0(x) &= \frac{3}{2}x^3 - \frac{5}{2}x^2 + 1 \\ f_1(x) &= -\frac{3}{2}x^3 + 2x^2 + \frac{1}{2}x \\ f_2(x) &= -\frac{1}{2}x^3 + x^2 - \frac{1}{2}x \\ f_3(x) &= \frac{1}{2}x^3 - \frac{1}{2}x^2 \end{aligned} \tag{8}$$

According to the observations of the transformed polynomials, here following values are calculated: x^2, x^3, $\frac{3}{2}x^3$, $-\frac{1}{2}x^3$ and $\frac{1}{2}x$. This strategy reduces the number of operations even more than using *Horner's rule*. Table 1 and 2 show the coefficient of the transformed polynomials for cubic and quintic order interpolation.

4 PERFORMANCE ANALYSIS

Table 3 shows the number of operations for the polynomial evaluation for the classical 1-D cubic convolution. Table 4 shows the results for arbitrary n-th order interpolation. As traditional piecewise n-th order convolution is separable (cascaded convolution), the evaluation of N-D convolution takes $N\times$ 1-D convolution time.
For 1-D cubic convolution the new method only needs 16 instead of 26 operations, which means, that the proposed method manages to calculate the same values with only 61.5% of operations.
When assessing the whole interpolation task, we must consider that the evaluation of the interpolation weights is one task, while the second task is to convolve the N-D function with these weights. The complexity of this convolution is given by $(n + 1)^N$ additions, multiplications and memory accesses. This basically means, that the convolution dominates over the polynomial evaluation in higher dimensions than 2.
Therefore the achieved speedup, by transforming the polynomials to $[0, 1]$, for the complete interpolation task is higher for lower dimension interpolation, namely 1-D and 2-D.

4.1 Real world experiments

A program was implemented in C++ to analyse the behavior of the new polynomial evaluation in a real world scenario. All calculation are done on a Pentium III, 1 GHz. Of course the kernel functions could be precalculated using lookup tables. To keep the accuracy at a high level we choose 10000 grid points per unit length according to [Ostun97]. Note that memory access is quite slow, therefore the lookup table approach is slower than direct calculation for the cubic case. To get an average value we looped over the calculation for 100 million times, which leads to the following results in 2-D interpolation:

Evaluation of the polynomials:

using lookup tables	18988 ms
classical method	13790 ms
new method	7962 ms

Whole interpolation task:

using lookup tables	51194 ms
classical method	45996 ms
new method	40168 ms

The new polynomial evaluation only takes 57.7% of computation time in comparison to the classical version, which matches quite well with the theoretical value of 61.5%. The whole 2-D cubic convolution task needs 87.3% in comparison to the classical version.

5 CONCLUSION

This paper presents a novel method for efficient implementation of cubic and higher order interpolation. The proposed polynomial evaluation needs only about 60% of computation time for cubic and about 65% for higher order polynomials in comparison to the standard method. The whole image interpolation process is speeded up and takes only about 87% for cubic convolution in comparison to the standard method.

6 ACKNOWLEDGEMENTS

Thanks to Christopher Zach for mathematical support.

A PROOFS

Lemma 1: The lowest order coefficients a_{i0} are 0 for all polynomials $f_i(x)$, except for the first polynomial $f_0(x)$ where the coefficient is 1.
Proof: The first polynomial $f_0(0) = 1 \Rightarrow a_{00} = 1$. For all other polynomials $f_i(0) = 0$ therefore $a_{i0} = 0$. $\Box$

Lemma 2: The highest order coefficients a_{in} of pairwise symmetrical polynomials, $f_i(x)$ and $f_{i+1}(x)$, i odd, have the same values with opposite sign.
Proof: The symmetrical polynomials called $f(x) := f_i(x) = \sum_{j=0}^{n} a_j x^j$ and $g(x) := f_{i+1}(x) = \sum_{j=0}^{n} b_j x^j$ have the property that $f(x) = g(1-x)$. This leads to:

$$f(x) = a_n x^n \sum_{j=1}^{n} a_j x^j = b_n (1-x)^n \sum_{j=1}^{n} b_j (1-x)^j$$

By using factor comparison of a_n and b_n, the coefficient for x^n is given by $a_n x^n = b_n(-x)^n = (-1)^n b_n x^n$. Because n is odd $(-1)^n = -1$ and this yields $a_n = -b_n$. $\Box$

Lemma 3: All coefficients of order $j \leq n-2$ of pairwise non symmetrical polynomials have same the coefficients of odd order and the same coefficients with opposite sign for even order.
Proof: f_i and f_{i+1}, $i \geq 1$ and i odd: Because of the continuous and smooth interpolation constraints $f_i^{(l)}(0) = (-1)^l f_{i+1}^{(l)}(0)$, which yields $a_{il} = (-1)^l a_{i+1l}$ for $l \leq n-2$. $\Box$

Lemma 4: All coefficients a_{nj} with $j \leq n-2$ of the polynomial $f_n(x)$ are 0.
Proof: Because of the continuous and smooth interpolation constraints the last polynomial $f_n(x)$ must be continuous to the x-axis, therefore $f_n^{(l)}(0) = 0$ for $l \leq n-2$ which directly yields $a_{nj} = 0$ for $j =\leq n-2$. $\Box$

Lemma 5: All coefficients a_{0j} with odd order j with $j \leq n-2$ are 0 for the polynomial $f_0(x)$.
Proof: $f_0(x)$ is defined as (equation 1) $f_0(x) = \sum_{j=0}^{n} a_{0j}|x|^j$ and is l-times continuous at $x = 0$. So $\lim_{x\to 0-} f^{(l)}(x) = \lim_{x \to 0+} f^{(l)}(x)$, which gives $\lim_{x\to 0-} f^{(l)}(0) = a_{0l} \overset{!}{=} \lim_{x\to 0+} f^{(l)}(0) = (-1)^l a_{0l}$. Therefore for all $j \leq n-2$ and j odd: $a_{0j} = 0$. $\Box$

References

[Keys81] R. G. Keys. Cubic convolution interpolation for digital image processing. *IEEE*

	a_{i3}	a_{i2}	a_{i1}	a_{i0}
f_0	$3/2$	$-5/2$	0	1
f_1	$-3/2$	2	$1/2$	0
f_2	$-1/2$	1	$1/2$	0
f_3	$1/2$	$-1/2$	0	0

Table 1: Coefficients of the transformed cubic convolution kernels

	a_{i5}	a_{i4}	a_{i3}	a_{i2}	a_{i1}	a_{i0}
f_0	$-27/32$	$63/32$	0	$-17/8$	0	1
f_1	$27/32$	$-9/4$	$9/16$	$5/4$	$19/32$	0
f_2	$13/64$	$-19/64$	$-9/16$	$5/4$	$-19/32$	0
f_3	$-13/64$	$23/32$	$-9/32$	$-3/16$	$-3/64$	0
f_4	$3/64$	$-3/16$	$9/32$	$-3/16$	$3/64$	0
f_5	$-3/64$	$3/64$	0	0	0	0

Table 2: Coefficients of the transformed quintic convolution kernels

	classical method		new method	
	$*$	$+$	$*$	$+$
calculation of distances	0	4	0	1
polynomial evaluation	12	10	8	7
overall	12	14	8	8

Table 3: Number of needed operations for cubic convolution polynomial evaluation

	classical method		new method	
	$*$	$+$	$*$	$+$
calculation of distances	0	$n+1$	0	1
polynomial evaluation	$n(n+1)$	$n(n+1)$	$n(n+3)/2$	$n^2-3/2n+5/2$
overall	$n(n+1)$	$(n+1)^2$	$n(n+3)/2$	$n^2-3/2n+7/2$

Table 4: Maximal number of needed operations for n-th order convolution polynomial evaluation

	classical method		new method		
order	$*$	$+$	$*$	$+$	factor
3	12	16	9	8	0.6071
5	30	36	20	21	0.6212
7	56	64	35	42	0.6417
9	90	100	54	71	0.6579

Table 5: Maximal number of needed operations for $3^{rd}-9^{th}$ order convolution polynomial evaluation. It could be possible that some coefficients are 0 or 1 by chance which decreases the number of needed operations.

Transactions on Acoustics, Speech, and Signal Processing, 29(6):1153–1160, 1981.

[Lehma99] T.M. Lehmann, C. Gönner, and K. Spitzer. Survey: interpolation methods in medical image processing. *IEEE Transactions on Medical Imaging*, 18(11):1049 – 1075, November 1999.

[Meije99] E. Meijering, K. Zuidervel, and M. Viergever. Image reconstruction with symmetrical piecewise nth-order polynomial kernels. *IEEE Transactions on Image Processing*, 8(2):192–201, February 1999.

[Meije01] E. Meijering, W. Niessen, and M. Viergever. Quantitative evaluation of convolution-based methods for medical image interpolation. *Medical Image Analysis*, 5:111–126, 2001.

[Ostun97] J. L. Ostuni, A. K. S. Santha, V. S. Mattay, D. R. Weinberger, R. L. Levin, and J. A. Frank. Analysis of interpolation effects in the reslicing of functional MR images. *J. Comp. Assisted Tomogr.*, 21(5):803810, 1997.

[Rifma73] S. S. Rifman. Digital rectification of ETRS multispectral imagery. *Proc. Symp. Significant Results Obtained form the Earth Resources Technology Satellite-1*, Section B, NASA SP-327(1):1131–1142, 1973.

[Unser99] Michael Unser. Splines: A perfect fit for signal and image processing. *IEEE Signal Processing Magazine*, 16(6):22–38, November 1999.