Published in IET Image Processing
Received on 23rd April 2008
Revised on 11th November 2008
doi: 10.1049/iet-ipr.2008.0172

ISSN 1751-9659

Fast colour space transformations using minimax approximations

M.Emre. Celebi[1] *H.A. Kingravi*[2] *F. Celiker*[3]

[1]*Department of Computer Science, Louisiana State University, Shreveport, LA, USA*
[2]*Department of Computer Science, Georgia Institute of Technology, Atlanta, GA, USA*
[3]*Department of Mathematics, Wayne State University, Detroit, MI, USA*
E-mail: ecelebi@lsus.edu

Abstract: Colour space transformations are frequently used in image processing, graphics and visualisation applications. In many cases, these transformations are complex non-linear functions, which prohibit their use in time-critical applications. A new approach called minimax approximations for colour space transformations (MACT) is presented. The authors demonstrate MACT on three commonly used colour space transformations. Extensive experiments on a large and diverse image set and comparisons with well-known multidimensional look-up table interpolation methods show that MACT achieves an excellent balance among four criteria: ease of implementation, memory usage, accuracy and computational speed.

1 Introduction

Colour space transformations are commonly used in various image processing, graphics and visualisation tasks for decoupling luminance and chromaticity information, ensuring approximate perceptual uniformity or achieving invariance to different imaging conditions such as viewing direction, illumination intensity and highlights. Since colour devices are usually provided with direct RGB signal input and output, the RGB colour space is generally the source in these transformations.

In many cases, colour space transformations are complex non-linear functions that prohibit their use in time-critical applications. In this paper, we present a new approach called minimax approximations for colour space transformations (MACT). We demonstrate MACT on transformations between the RGB space and three popular colour spaces: CIELAB, Hue–saturation–intensity (HSI) and spherical coordinate transform (SCT). There are various reasons behind the choice of these particular spaces. First, all three spaces decouple luminance and chromaticity information, which makes them suitable for tasks including enhancement [1, 2], rendering [3], noise removal [4, 5], segmentation [6–8] and object recognition [9, 10]. Second, an important feature of CIELAB is its approximate perceptual uniformity, an essential requirement for expressing colour differences in applications such as colour quantisation [11], mesh denoising [12] and maximum contrast colour set design [13]. Third, HSI and SCT are designed to match human intuition, which makes them useful for manual colour selection [14]. Last, the first two components of HSI are shown to be invariant to viewing direction, surface orientation, illumination direction and illumination intensity [10].

The rest of the paper is organised as follows. Section 2 gives the related work. Section 3 presents the use of minimax approximation theory in speeding up colour space transformations and the experimental results. Finally, Section 4 gives the conclusions.

2 Related work

Traditionally, colour space transformations in digital imaging systems have been implemented using look-up tables (LUTs) that require some form of multidimensional interpolation. The most commonly used 3D LUT interpolation methods are trilinear, prism, pyramidal and tetrahedral [15]. These methods are explained in Section 3.5.3. In this section, we briefly review the interpolation methods that appear

less frequently in the literature as well as some alternative approaches.

Chang *et al.* [16] developed a method called sequential linear interpolation (SLI) that uses a partially separable grid structure, which allows the allocation of more grid points to the regions where the function to be interpolated is more non-linear. This scheme often results in more accurate transformations at the expense of increased computational cost when compared to trilinear interpolation. Kacker *et al.* [17] proposed a wavelet-based method that uses a multiscale grid structure. This method is shown to achieve lower maximum error but higher average error when compared to SLI. Gupta and Gray [18] presented a method called maximum entropy estimation, which is a generalisation of tetrahedral interpolation. This method is shown to be more accurate than tetrahedral interpolation in general while being at best only half as fast. Hemingway [19] described a method based on n-simplexes that is slightly faster than tetrahedral interpolation. However, the accuracy of the method is not discussed in the paper.

Neural networks have been applied to the colour space transformation problem in a number of studies [20–23]. They have the advantages of being more flexible and requiring less memory. However, they require training and parameter tuning, and are prone to overfitting [18].

3 Approximate colour space transformations

3.1 Overview of minimax approximation theory

Given a function f, we would like to approximate it by another function g so that the error (ε) between them over a given interval is arbitrarily small. The existence of such approximations is stated by the following theorem:

Theorem 3.1 (Weierstrass): Let f be a continuous real-valued function defined on $[a, b]$, that is, $f \in C[a, b]$. Then $\forall \varepsilon > 0$, there exists a polynomial P such that $\|f - P\| < \varepsilon$, that is, $\forall x \in [a, b]$, $|f(x) - P(x)| < \varepsilon$.

This is commonly known as the minimax approximation to a function. It differs from other methods, for example, least-squares approximations, in that it minimises the maximum error (ε) rather than the average error

$$\varepsilon = \max_{x \in [a,b]} |f(x) - P(x)| \tag{1}$$

A similar theorem establishes the existence of a rational variant of this method [24]. Let $n \geq 0$ be a natural number and let

$$P_n([a, b]) = \{a_0 + a_1 x + \cdots + a_n x^n : x \in [a, b], \; a_i \in \mathbb{R}, \; i = 0, 1, \ldots, n\} \tag{2}$$

be the set of all polynomials of degree less than or equal to n. The set of irreducible rational functions, $R_m^n([a, b])$, is defined as

$$R_m^n([a, b]) = \left\{ \frac{p(x)}{q(x)} : p(x) \in P_n([a, b]), \; q(x) \in P_m([a, b]) \right\} \tag{3}$$

where p and q have no common factors. Then [24]

Theorem 3.2: For each function $f \in C[a, b]$, there exists at least one best rational approximation from the class $R_m^n([a, b])$.

This theorem states the existence of a rational approximation $r^* \in R_m^n([a, b])$ to a function $f \in C[a, b]$ that is optimal in the Chebyshev sense

$$\max_{x \in [a,b]} |f(x) - r^*(x)| = \text{dist}(f, R_m^n) \tag{4}$$

where $\text{dist}(f, R_m^n)$ denotes the distance between f and $R_m^n([a, b])$ with respect to some norm, in our case, the Chebyshev (maximum) norm. Regarding the choice between a polynomial and a rational approximant, it can be said that certain functions can be approximated more accurately by rationals than by polynomials. Jean-Michel Muller explains this phenomenon as follows: 'It seems quite difficult to predict if a given function will be much better approximated by rational functions than by polynomials. It makes sense to think that functions that have a behaviour that is "highly nonpolynomial" (finite limits at $\pm\infty$, poles, infinite derivatives, ...) will be poorly approximated by polynomials.' [25].

In this study, the Remez exchange algorithm is used to calculate the minimax approximations. The reader is referred to [24, 25] for more information on the theory of minimax approximations and [26] for the implementation details of the Remez algorithm.

3.2 CIELAB colour space

3.2.1 Colour space description: CIELAB is an approximately uniform colour space standardised by the Commission Internationale de l'Eclairage (CIE) in 1976 [27]. The L^* component represents the lightness, whereas a^* and b^* represent the chromaticity. The transformation between the RGB and CIELAB colour spaces is comprised of three steps: (i) conversion from non-linear $R'G'B'$ to linear RGB, (ii) conversion from linear RGB to CIEXYZ and (iii) conversion from CIEXYZ to CIELAB.

In order to convert the non-linear $R'G'B'$ values to linear RGB ones, inverse gamma correction (ITU-R BT.709) is

performed

$$k' \in \left\{\frac{R'}{255}, \frac{G'}{255}, \frac{B'}{255}\right\}$$

$$k \in \{R, G, B\} = \begin{cases} \dfrac{k'}{4.5}, & 0 \le k' < 0.081 \\ \left(\dfrac{k' + 0.099}{1.099}\right)^{1/0.45}, & 0.081 \le k' \le 1 \end{cases} \tag{5}$$

The conversion from the linear RGB to CIEXYZ (ITU-R BT.709) is given by

$$\begin{aligned} X &= 0.412391R + 0.357584G + 0.180481B \\ Y &= 0.212639R + 0.715169G + 0.072192B \\ Z &= 0.019331R + 0.119195G + 0.950532B \end{aligned} \tag{6}$$

Finally, the conversion from CIEXYZ to CIELAB is given by

$$\begin{aligned} L^* &= 116f(Y/Y_0) - 16 \\ a^* &= 500[f(X/X_0) - f(Y/Y_0)] \\ b^* &= 200[f(Y/Y_0) - f(Z/Z_0)] \end{aligned} \tag{7}$$

$$f(t) = \begin{cases} t^{1/3}, & t > 0.008856 \\ 7.787t + 16/116, & t \le 0.008856 \end{cases}$$

Here, X_0, Y_0 and Z_0 are the tristimulus values of the reference white. For the illuminant D65 these are

$$\begin{aligned} X_0 &= 0.950456 \\ Y_0 &= 1.0 \\ Z_0 &= 1.089058 \end{aligned} \tag{8}$$

The distance between two pixels $\boldsymbol{x}$ and $\boldsymbol{y}$ in the CIELAB space is given by

$$D_{\text{CIELAB}}(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(L_{\boldsymbol{x}}^* - L_{\boldsymbol{y}}^*)^2 + (a_{\boldsymbol{x}}^* - a_{\boldsymbol{y}}^*)^2 + (b_{\boldsymbol{x}}^* - b_{\boldsymbol{y}}^*)^2} \tag{9}$$

3.2.2 Approximation of the cube-root function:

Equations (5) and (6) can be implemented efficiently using LUTs, as commonly seen in the literature [15]. The cube-root function (cbrt) in (7) is the main factor that influences the computational time of the transformation. The probabilities of calling this function can be calculated from an image that contains every possible colour in the 24-bit RGB space (see Section 3.5.1)

$$\begin{aligned} P(X/X_0 > 0.008856) &= 0.999626 \\ P(Y/Y_0 > 0.008856) &= 0.999165 \\ P(Z/Z_0 > 0.008856) &= 0.997184 \end{aligned} \tag{10}$$

These high values suggest that we can accelerate the transformation substantially if we can devise a fast approximation for the cube-root function (see Fig. 1). Table 1 shows the coefficients of the minimax polynomials of various degrees. Here, n and $\varepsilon_{\max}^{\text{A}}$ represent the degree of the polynomial and error of the minimax approximation, respectively.

It can be seen that the error values are quite high and as the approximation degree is increased, the accuracy does not improve significantly. This suggests that rational functions might be better suited for this approximation task. Table 2 shows the coefficients of minimax rationals of various degrees. Here, each pair of adjacent rows corresponds to a rational function of a particular degree (n, m) in which the first and second rows represent the numerator and denominator, respectively. It can be seen that minimax rationals can accurately represent the cube-root function.

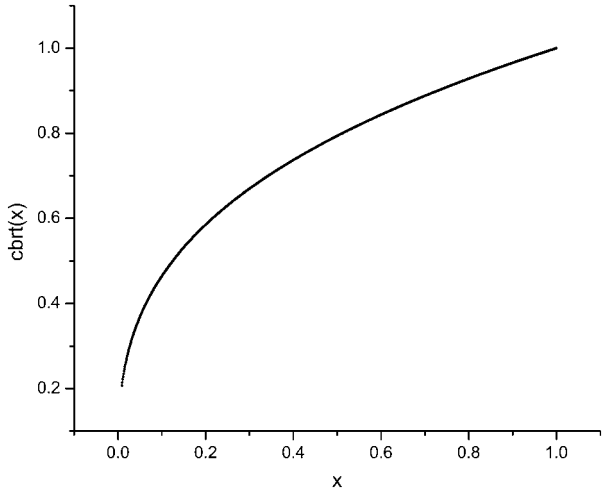

Figure 1 *Cube-root function in the interval [0.008856, 1]*

Table 1 Minimax polynomials for the cube-root function

n	$\varepsilon_{\max}^{\text{A}}$	a_0	a_1	a_2	a_3	a_4	a_5
2	1.271154e − 01	1.268979e − 01	2.393873	−1.647669			
3	9.787829e − 02	9.787826e − 02	4.057495	−7.388864	4.331370		
4	8.111150e − 02	8.111133e − 02	5.926004	−2.017165e + 01	2.833070e + 01	−1.324728e + 01	
5	7.002956e − 02	7.002910e − 02	7.961214	−4.329352e + 01	1.063182e + 02	−1.135685e + 02	4.358268e + 01

doi: 10.1049/iet-ipr.2008.0172

Table 2 Minimax rationals for the cube-root function

n	m	ε^{A}_{max}	a_0	a_1	a_2	a_3	a_4
2	2	2.060996e − 03	6.309655e − 03	5.785782e − 01	1.591005		
			4.482646e − 02	1.175862	9.596879e − 01		
2	3	7.210231e − 04	2.500705e − 03	3.447113e − 01	1.942708		
			1.978701e − 02	8.542797e − 01	1.664540	−2.503267e − 01	
3	2	5.931593e − 04	1.776519e − 03	2.632323e − 01	1.751297	3.836709e − 01	
			1.432256e − 02	6.779998e − 01	1.706260		
2	4	3.107735e − 04	1.317899e − 03	2.390113e − 01	2.099395		
			1.124254e − 02	6.726679e − 01	2.184656	−7.511150e − 01	2.229717e − 01
3	3	1.858694e − 04	4.370889e − 04	9.526952e − 02	1.252009	1.302733	
			3.912364e − 03	2.954084e − 01	1.717143	6.343408e − 01	
4	2	2.334688e − 04	7.589302e − 04	1.519784e − 01	1.663584	8.368075e − 01	−1.657269e − 01
			6.644723e − 03	4.506424e − 01	2.030622		
3	4	8.539863e − 05	1.683667e − 04	4.667675e − 02	9.106812e − 01	1.810577	
			1.610864e − 03	1.617974e − 01	1.494070	1.218468	−1.079451e − 01
4	3	8.052920e − 05	1.349673e − 04	3.832079e − 02	7.870174e − 01	1.799062	2.071170e − 01
			1.299250e − 03	1.345420e − 01	1.330358	1.365369	
4	4	3.856930e − 05	3.927283e − 05	1.392318e − 02	4.114739e − 01	1.734853	8.679223e − 01
			4.022100e − 04	5.414536e − 02	8.221526e − 01	1.800167	3.513617e − 01

3.3 HSI colour space

3.3.1 Colour space description: HSI is an intuitive alternative to the RGB space [27]. It uses approximately cylindrical coordinates, and is a non-linear deformation of the RGB colour cube. The hue H is a function of the angle in the polar coordinate system and describes a pure colour. The saturation S is proportional to radial distance and denotes the purity of a colour. Finally, the intensity I is the distance along the axis perpendicular to the polar coordinate plane and represents the brightness.

The transformation between RGB and HSI is given by

$$
\begin{aligned}
&H = \arccos\left[\frac{0.5(R-G+R-B)}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right]\\
&\text{if } (B > G)\\
&\quad H = 2\pi - H\\
&\quad S = 1 - 3\min(R, G, B)/(R+G+B)\\
&\quad I = (R+G+B)/3
\end{aligned}
\tag{11}
$$

where arccos denotes the inverse cosine function. Kender [28] proposed a fast version of (11) that gives numerically identical results. This transformation involves fewer multiplications and no square-root operation

$$
\begin{aligned}
&\text{if } R > B \quad \text{and} \quad G > B\\
&H = \frac{\pi}{3} + \arctan\left[\frac{\sqrt{3}(G-R)}{G-B+R-B}\right]\\
&\text{else if } G > R\\
&H = \pi + \arctan\left[\frac{\sqrt{3}(B-G)}{B-R+G-R}\right]\\
&\text{else if } B > G\\
&H = \frac{5\pi}{3} + \arctan\left[\frac{\sqrt{3}(R-B)}{R-G+B-G}\right]\\
&\text{else if } R > B\\
&H = 0\\
&\text{else}\\
&H = \text{undefined}
\end{aligned}
\tag{12}
$$

where arctan denotes the inverse tangent function. The distance between two pixels $\boldsymbol{x}$ and $\boldsymbol{y}$ in the HSI space is

given by

$$D_{\mathrm{HSI}}(x, y) = \sqrt{s_x^2 + s_y^2 - 2 s_x s_y \cos\,\theta + (i_x - i_y)^2}$$
$$\theta = \begin{cases} |h_x - h_y| & \text{if } |h_x - h_y| \le \pi \\ 2\pi - |h_x - h_y| & \text{otherwise} \end{cases} \quad (13)$$

3.3.2 Approximation of the inverse tangent function: We decided to approximate Kender's transformation (12) rather than the original one (11) because of several reasons. First, these two give identical results. Second, (12) is computationally cheaper than (11). Third, as can be seen in Section 3.4, the inverse tangent function in (12) is easier to approximate when compared to the inverse cosine function in (11).

In (12), all of the cases involve multiplication by the constant $\sqrt{3}$. Therefore a multiplication operation can be avoided by approximating arctan $(\sqrt{3}x)$. Note that the argument of this function can also be negative. This can be handled using the following identity

$$\arctan(x) = -\arctan(-x) \quad (14)$$

In (12), the inverse tangent function receives its arguments from the interval $[-1.0, 1.0)$. [This holds prior to the multiplication with $\sqrt{3}$.] Fig. 2 shows a plot of the function in the second-half of this interval.

Owing to its highly linear behaviour, this function can be accurately approximated by low-order polynomials. Table 3 shows the coefficients of the minimax polynomials of various degrees (n).

3.4 SCT colour space

3.4.1 Colour space description: The SCT is defined as [29]

$$L = \sqrt{R^2 + G^2 + B^2}$$
$$\angle A = \arccos(B/L)$$
$$\angle B = \arccos\left(\frac{R}{L\,\sin(\angle A)}\right) \quad (15)$$

where L represents the luminance and angles $\angle A$ and $\angle B$ represent the chromaticity. The expression for $\angle B$ can be simplified using trigonometric manipulations

$$\angle B = \arctan(G/R) \quad (16)$$

This formulation is computationally advantageous in that it avoids a multiplication and a sine operation.

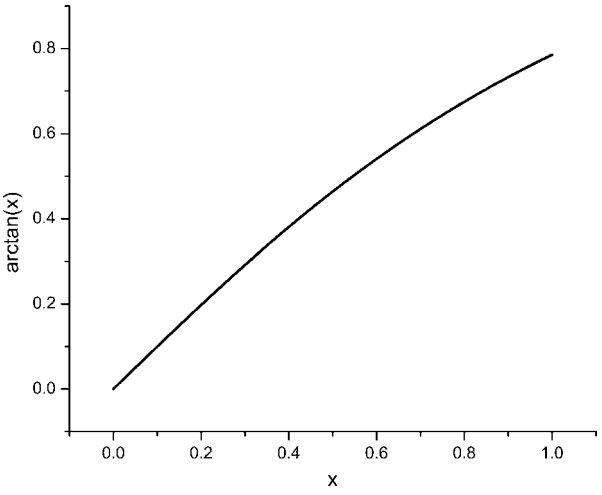

Figure 2 *Inverse tangent function in the interval [0, 1]*

Unfortunately, it is not easy to define a perceptual distance function in SCT. Although various formulae have been developed to calculate the distance between two points lying on the same spherical surface, these cannot be used in SCT. This is because pixels of different brightness in this space lie on different spherical shells. Therefore we decided to use the following alternative approach. Given two pixels in SCT, the inverse transformation (17) is applied to switch back to the RGB space. The pixels are then converted to CIELAB so that the distance between them can be calculated using (9). This indirect method of distance calculation is likely to introduce additional errors. However, minimax approximations for the elementary functions involved in (15) and (16) can be devised to obtain an arbitrarily accurate approximate transformation.

$$R = L\,\sin(\angle A)\cos(\angle B)$$
$$G = L\,\sin(\angle A)\sin(\angle B) \quad (17)$$
$$B = L\,\cos(\angle A)$$

3.4.2 Approximation of the inverse cosine function: In the expression for $\angle A$, the inverse cosine

Table 3 Minimax polynomials for the inverse tangent function

n	$\varepsilon_{\max}^{A}$	a_0	a_1	a_2	a_3	a_4	a_5
2	6.907910e − 03	5.959793e − 03	1.782975	7.497879e − 01			
3	3.654156e − 03	−3.654076e − 03	1.884080	−9.805583e − 01	1.430580e − 01		
4	1.286371e − 03	−1.286369e − 03	1.796716	−4.958969e − 01	−6.927404e − 01	4.421541e − 01	
5	1.801311e − 04	−1.801283e − 04	1.739333	−2.039848e − 02	−2.065512	2.052837	−6.591729e − 01

doi: 10.1049/iet-ipr.2008.0172

function receives its arguments from the interval [0, 1]. Fig. 3 shows a plot of this function.

Unfortunately, approximating the inverse cosine function in this interval is not easy because of its behaviour near 1. This can be circumvented using the following numerically more stable identity for $x \geq 0.5$

$$\arccos(x) = 2\arcsin\left(\sqrt{0.5(1-x)}\right) \tag{18}$$

In (18), the inverse sine function (arcsin) receives its arguments from the interval [0, 0.5]. Fig. 4 shows a plot of this function. In order to avoid two multiplication operations, the following function can be approximated instead

$$\begin{aligned} y &= \sqrt{1-x} \\ \arccos(x) &= 2\arcsin(y/\sqrt{2}) \end{aligned} \tag{19}$$

In (19), the argument y falls into the interval $[0, 1/\sqrt{2}]$. Table 4 shows the coefficients of the minimax polynomials of various degrees (n).

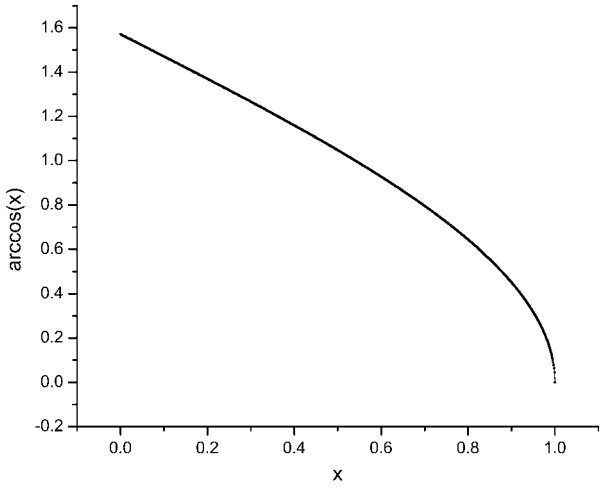

Figure 3 *Inverse cosine function in the interval [0, 1]*

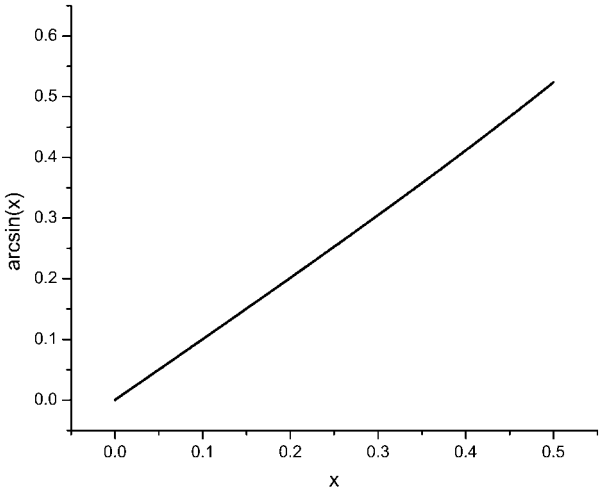

Figure 4 *Inverse sine function in the interval [0, 0.5]*

On the other hand, it can be seen from Fig. 3 that the inverse cosine function is highly linear in the interval [0, 0.5] and can be accurately approximated by polynomials. Table 5 shows the coefficients of the minimax polynomials of various degrees (m).

3.4.3 Approximation of the inverse tangent function: In (16), the inverse tangent function receives its arguments from the interval [0, 255]. Fig. 5 shows a plot of this function. Note that the degenerate cases, that is, $R = G = 0$ ($\angle B =$ undefined) and $R = 0$, $G > 0$ ($\angle B = \pi/2$), are not reflected in the plot.

It can be seen that the function exhibits high variability in this large interval. In order to obtain an accurate low-order polynomial approximation, the domain can be divided into two as follows

$$\begin{aligned} x &= G/R \\ \arctan(x) &= \begin{cases} f(x) = \arctan(x) & \text{if } G < R \\ g(x) = \pi/2 - \arctan(1/x) & \text{otherwise} \end{cases} \end{aligned} \tag{20}$$

Table 4 Minimax polynomials for the inverse sine function

n	ε^{A}_{max}	a_0	a_1	a_2	a_3	a_4	a_5
4	2.097814e − 05	2.097797e − 05	1.412840	1.429881e − 02	6.704361e − 02	6.909677e − 02	
5	2.370540e − 06	−2.370048e − 06	1.414434	−3.300037e − 03	1.354670e − 01	−3.994259e − 02	6.099502e − 02

Table 5 Minimax polynomials for the inverse cosine function

m	ε^{A}_{max}	a_0	a_1	a_2	a_3	a_4	a_5
4	1.048949e − 05	1.570786	−9.990285e − 01	−1.429899e − 02	−9.481335e − 02	−1.381942e − 01	
5	1.186403e − 06	1.570798	−1.000156	3.299810e − 03	−1.915780e − 01	7.988231e − 02	−1.725177e − 01

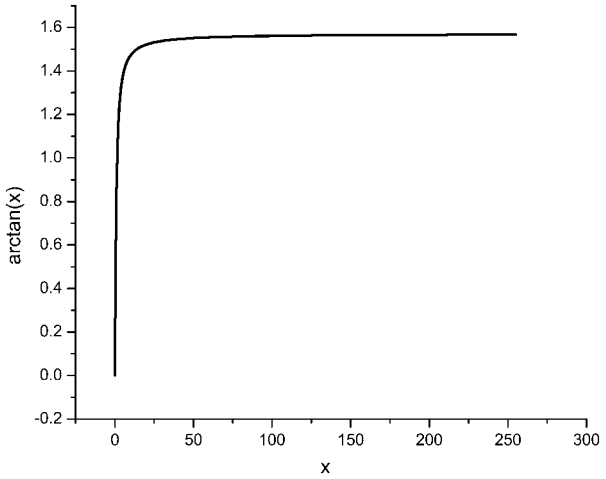

Figure 5 *Inverse tangent function in the interval [0, 255]*

The second part of (20) follows from

$$\arctan(x) = \pi/2 - \arctan(1/x) \quad \text{for } x > 0 \tag{21}$$

When $G < R$, the function f receives its arguments from the interval [0, 254/255]. On the other hand, when $R \leq G$ the inverse tangent function within g receives its arguments from the interval [1/255, 1]. Tables 6 and 7 show the coefficients of the minimax polynomials of various degrees (r) for the functions f and g, respectively.

As in the case of (10), the probabilities of calling the inverse sine and inverse cosine functions can be calculated from an image that contains every possible colour in the 24-bit RGB space

$$\begin{aligned} P_{\text{arcsin}} &= P(B/L > 0.5) = 0.557723 \\ P_{\text{arccos}} &= 1 - P_{\text{arcsin}} = 0.442277 \end{aligned} \tag{22}$$

In contrast, the inverse tangent function in (16) is almost always called (except for when $R = G = 0$ in which case $\angle B =$ undefined)

$$P_{\text{arctan}} = 1 - P(R = G = 0) = 1 - \frac{256}{256^3} = 0.999985 \tag{23}$$

Note that $\angle B = \pi/2$ when $R = 0$ and $G > 0$.

3.5 Experimental results

3.5.1 Calculation of the colour space transformation errors: In order to calculate the accuracy of the presented approximate colour space transformations, we used a 4096 × 4096 image (henceforth referred to as RGB16Million) [available at http://brucelindbloom.com/downloads/RGB16Million.tif.zip], which contains 16 777 216 unique colours, that is, every possible colour in the 24-bit RGB space. Tables 8–10 show the average execution time (in seconds) [programming language: C, compiler: gcc 3.4.4, CPU: Intel Pentium D 2.66 GHz] over 1000 identical runs and the average ($\varepsilon^{\text{T}}_{\text{avg}}$) and maximum ($\varepsilon^{\text{T}}_{\text{max}}$) transformation errors for CIELAB (9), HSI (13) and SCT [(9) and (17)], respectively. Note that the rows of Table 8 are sorted on $(n + m)$, since the execution time is proportional to the total degree of the polynomials in the rational. On the other hand, the rows of Table 10 are sorted on the degree of the inverse tangent approximation (r). This is because the probability of calling this function (23) is much higher than that of the other two elementary functions (22). Therefore the execution time is mainly influenced by the degree of the inverse tangent approximation.

We also performed the exact transformations on RGB16Million and calculated the average execution time over 100 runs. The results were 60.331, 13.485 and 27.599 s for CIELAB, HSI and SCT, respectively. Comparing these values with those given in Tables 8–10, we can see that the proposed approximations provide substantial computational savings.

3.5.2 Calculation of the computational gain values: It can be seen from Tables 8–10 that only marginal computational gains can be obtained using lower-order approximations without significantly compromising the accuracy of the transformation. Therefore in this section, we consider only the highest-order approximations for each transformation. However, lower-order approximations might be preferable depending on the application requirements.

In order to calculate the computational gain for each approximate transformation, a set of 100 high-quality RGB images was collected from the Internet. The set includes images of people, animals, plants, buildings, aerial maps, man-made objects, natural scenery, paintings, sketches, as well as scientific, biomedical, synthetic images and test images commonly used in the literature.

On each image, the exact transformations were performed 100 times and the average execution times were calculated. The same was done for the corresponding approximate transformations with 1000 runs. The computational gain is calculated as the ratio of the average execution times. Table 11 shows the statistics over the entire image set. It

Table 6 Minimax polynomials for the function f

r	$\varepsilon^{\text{A}}_{\text{max}}$	a_0	a_1	a_2	a_3	a_4	a_5
4	1.036515e − 04	−1.036508e − 04	1.003740	−1.773538e − 02	−3.390563e − 01	1.386796e − 01	
5	2.073939e − 05	2.073866e − 05	9.982666e − 01	2.352573e − 02	−4.506862e − 01	2.635050e − 01	−4.920822e − 02

IET Image Process., 2010, Vol. 4, Iss. 2, pp. 70–80
doi: 10.1049/iet-ipr.2008.0172

Table 7 Minimax polynomials for the function g

r	ε_{max}^{A}	a_0	a_1	a_2	a_3	a_4	a_5
4	1.051643e − 04	1.570917	−1.004004	1.885694e − 02	3.373159e − 01	−1.377930e − 01	
5	2.012104e − 05	1.570769	−9.981253e − 01	−2.440212e − 02	4.528921e − 01	−2.659181e − 01	5.016228e − 02

Table 8 Comparison of CIELAB approximations

n	m	Avg. time	ε_{avg}^{T}	ε_{max}^{T}
2	2	1.031872	0.787741	2.225399
2	3	1.134568	0.295577	0.774404
3	2	1.114776	0.239033	0.636929
2	4	1.221390	0.131411	0.331015
3	3	1.165900	0.061127	0.193565
4	2	1.197828	0.090846	0.247291
3	4	1.291526	0.018092	0.085046
4	3	1.291570	0.014927	0.079935
4	4	1.292872	0.003201	0.036481

Table 9 Comparison of HSI approximations

n	Avg. time	ε_{avg}^{T}	ε_{max}^{T}
2	0.865328	0.002523	0.007134
3	0.865575	0.001259	0.003693
4	0.866254	0.000448	0.001320
5	0.873251	0.000063	0.000190

can be seen that the computational gain observed in RGB16Million also applies to the case of real-world images.

3.5.3 Comparison with 3D LUT interpolation methods: In this section, we compare MACT with the most commonly used LUT interpolation methods, namely trilinear, prism, pyramidal and tetrahedral. These methods involve three steps: packing, extraction and interpolation [15]. Packing is a process that divides the domain of the source space and populates it with sample points to build the LUT. The extraction step aims at finding the location of the input pixel and extracting the colour values of the nearest lattice points. The last step is 3D interpolation, in which the input point and the extracted lattice points are used to calculate the destination colour specifications. Essentially, 3D interpolation is a repeated application of linear interpolation [15]

$$x = (1 - \alpha)x_0 + \alpha x_1 \tag{24}$$

where x_0 and x_1 are the spatial coordinates of the two known points and α is the interpolation coefficient. The 3D interpolation is the step in which the aforementioned LUT interpolation methods differ. Trilinear interpolation uses eight neighbouring lattices, whereas prism, pyramidal and tetrahedral interpolations use six, five and four neighbours, respectively. The greater the number of neighbouring lattices used in a method, the higher the computational requirements and accuracy. Since the formulations of these methods are mathematically involved, the interested reader is referred to the relevant literature [15].

Table 10 Comparison of SCT approximations

n	m	r	Avg. time	ε_{avg}^{T}	ε_{max}^{T}
4	4	4	1.330927	0.006596	0.024273
4	5	4	1.371811	0.006553	0.024273
5	4	4	1.377924	0.006266	0.022820
5	5	4	1.419315	0.006224	0.021810
4	4	5	1.427337	0.002139	0.008371
4	5	5	1.462627	0.002011	0.008371
5	4	5	1.471486	0.001381	0.005793
5	5	5	1.507812	0.001254	0.004543

Table 11 Computational gain statistics for each transformation

Transformation	Computational gain				
	Min	Max	Mean	Stdev	Median
CIELAB	8.761736	40.629135	35.575373	4.608590	37.192729
HSI	4.511637	16.460248	11.437106	2.439482	11.669715
SCT	4.493543	22.232897	15.782453	2.110820	16.162434

Table 12 Comparison of 3D LUT interpolation methods

Method	LUT size	Avg. time (standard)	Avg. time (caching)	ε^{T}_{avg}	ε^{T}_{max}
Trilinear	9^3	0.901673	0.619681	0.359677	5.409771
	17^3	0.908753	0.630465	0.100026	1.987040
	33^3	0.982880	0.716958	0.025089	0.653506
Prism	9^3	0.856652	0.609452	0.349128	5.409771
	17^3	0.860880	0.614681	0.095150	2.343905
	33^3	0.931349	0.677319	0.024172	0.826651
Pyramidal	9^3	0.850409	0.540481	0.319163	5.872591
	17^3	0.872813	0.566271	0.088116	2.909481
	33^3	0.940143	0.637049	0.022466	1.042665
Tetrahedral	9^3	0.689341	0.450413	0.284339	5.783921
	17^3	0.704121	0.486654	0.077870	2.702448
	33^3	0.773102	0.561076	0.020253	1.185788

We implemented two versions of each LUT method: standard and caching. The former is a direct implementation of the mathematical formulation, whereas the latter pre-computes the differences between neighbouring lattices and stores these in each node. Table 12 shows the average execution times on RGB16Million over 1000 runs and the transformation errors for CIELAB. It can be seen that even though these methods range from being 1.32 to 2.87 times faster than MACT, their accuracy is 18 to 161 times lower. In addition, these methods require extra storage of up to 10.7 MB. Note that LUT size affects only the accuracy of the transformation and the storage requirements and, in theory, it should not affect the computational time. However, it can be observed from the table that as the LUT size is increased, the computational time increases as well. This is most likely due to the limited size of the processor cache.

Since HSI and SCT include angular components (H component in HSI, $\angle A$ and $\angle B$ components in SCT), 3D interpolation in these colour spaces involves the interpolation of angular (circular) data given by [30]

$$\theta = \arctan\left(\frac{(1-\alpha)\sin(\theta_0) + \alpha\,\sin(\theta_1)}{(1-\alpha)\cos(\theta_0) + \alpha\,\cos(\theta_1)}\right) \qquad (25)$$

where θ_0 and θ_1 are the two known angles and α is the interpolation coefficient. Owing to the trigonometric and inverse trigonometric functions in (25), it turns out that 3D interpolation in these spaces is computationally more expensive than the original transformations given in (11) and (15). For example, trilinear interpolation in HSI takes about 84 s on RGB16Million ($\varepsilon^{T}_{avg} = 0.000777$, $\varepsilon^{T}_{max} = 1.086255$).

Note that even if the angular data were quantised into say, 360 steps, pre-computing (25) would not be possible since the value of the interpolation coefficient α is not known a priori. In addition, the inverse tangent function cannot be approximated using a minimax polynomial since its argument is not bounded. In short, the expensive interpolation operation offsets the computational advantage of 3D LUT interpolation in angular colour spaces.

4 Conclusions

In this paper we proposed MACT, a novel approach to speed up colour space transformations based on minimax approximations. Advantages of MACT include ease of implementation, negligible memory requirements, extremely good accuracy and very high computational speed. Comparisons with commonly used 3D LUT interpolation methods revealed that MACT yields significantly more accurate transformations at the expense of slightly higher computational requirements. Although MACT was applied to three particular colour space transformations, it can easily be adapted to other transformations, for example, RGB to CIELUV transformation, that involve computationally expensive mathematical functions.

Implementations of the fast colour space transformations described in this paper will be made publicly available at http://sourceforge.net/projects/fourier-ipal.

5 Acknowledgments

This publication was made possible by a grant from The Louisiana Board of Regents (LEQSF2008-11-RD-A-12). The authors are grateful to the anonymous reviewers for

doi: 10.1049/iet-ipr.2008.0172

their valuable comments and to Bruce Lindbloom for providing the RGB16Million image.

6 References

[1] KAO W.-C., CHEN Y.-J.: 'Multistage bilateral noise filtering and edge detection for color image enhancement', *IEEE Trans. Consum. Electron.*, 2005, **51**, (4), pp. 1346–1351

[2] PITAS I., KINIKLIS P.: 'Multichannel techniques in color image enhancement and modeling', *IEEE Trans. Image Process.*, 1996, **5**, (1), pp. 168–171

[3] EBERT D.S., MORRIS C.J., RHEINGANS P., YOO T.S.: 'Designing effective transfer functions for volume rendering from photographic volumes', *IEEE Trans. Vis. Comput. Graphics*, 2002, **8**, (2), pp. 183–197

[4] JIN L., LI D.: 'A switching vector median filter based on the CIELAB color space for color image restoration', *Signal Process.*, 2007, **87**, (6), pp. 1345–1354

[5] CELEBI M.E., KINGRAVI H.A., UDDIN B., ASLANDOGAN Y.A.: 'Fast switching filter for impulsive noise removal from color images', *J. Imaging Sci. Technol.*, 2007, **51**, (2), pp. 155–165

[6] COMANICIU D., MEER P.: 'Mean shift: a robust approach toward feature space analysis', *IEEE Trans. Pattern Anal. Mach. Intell.*, 2002, **24**, (5), pp. 603–619

[7] PEREZ F., KOCH C.: 'Toward color image segmentation in analog VLSI: algorithm and hardware', *Int. J. Comput. Vis.*, 1994, **12**, (1), pp. 17–42

[8] MINTEN B.W., MURPHY R.R., HYAMS J., MICIRE M.: 'Low-order-complexity vision-based docking', *IEEE Trans. Robot. Autom.*, 2001, **17**, (6), pp. 922–930

[9] CORBALÁN M., MILLÁN M.S., YZUEL M.J.: 'Color pattern recognition with CIELAB coordinates', *Opt. Engng.*, 2001, **41**, (1), pp. 130–138

[10] GEVERS T., SMEULDERS A.W.M.: 'Color based object recognition', *Pattern Recognit.*, 1999, **32**, (3), pp. 453–464

[11] WU X.: 'Color quantization by dynamic programming and principal analysis', *ACM Trans. Graph.*, 1992, **11**, (4), pp. 348–372

[12] FLEISHMAN S., DRORI I., COHEN-OR D.: 'Bilateral mesh denoising', *ACM Trans. Graph.*, 2003, **22**, (3), pp. 950–953

[13] GLASBEY C., VAN DER HEIJDEN G., TOH V., GRAY A.: 'Colour displays for categorical images', *Color Res. Appl.*, 2007, **32**, (4), pp. 304–309

[14] HSIAO S.-W., CHIU F.-Y., HSU H.-Y.: 'A computer-assisted colour selection system based on aesthetic measure for colour harmony and fuzzy logic theory', *Color Res. Appl.*, 2008, **33**, (5), pp. 411–423

[15] KANG H.R.: 'Computational color technology' (SPIE Press, 2006)

[16] CHANG J.Z., ALLEBACH J.P., BOUMAN C.A.: 'Sequential linear interpolation of multidimensional functions', *IEEE Trans. Image Process.*, 1997, **6**, (9), pp. 1231–1245

[17] KACKER D., AGAR A.U., ALLEBACH J.P., LUCIER B.J.: 'Wavelet decomposition based representation of nonlinear color transformations and comparison with sequential linear interpolation'. Proc. IEEE Int. Conf. Image Processing, 1998, pp. 186–190

[18] GUPTA M.R., GRAY R.M.: 'Color conversions using maximum entropy estimation'. Proc. IEEE Int. Conf. Image Processing, 2001, pp. 118–121

[19] HEMINGWAY P.: '*n*-Simplex interpolation'. Tech. Rep. HPL-2002-320, HP, 2002. Available at http://www.hpl.hp.com/techreports/2002/HPL-2002-320.html

[20] KANG H.R., ANDERSON P.G.: 'Neural network applications to the color scanner and printer calibrations', *J. Electron. Imaging*, 1992, **1**, (2), pp. 125–135

[21] TOMINAGA S.: 'Color notation conversion by neural networks', *Color Res. Appl.*, 1993, **18**, (4), pp. 253–259

[22] USUI S., ARAI Y., NAKAUCHI S.: 'Neural networks for device-independent digital color imaging', *Inf. Sci.*, 2000, **123**, (1/2), pp. 115–125

[23] VRHEL M.J.: 'Approximation of color characterization MLUTS with artificial neural networks'. Proc. IEEE Int. Conf. Image Processing, 2003, pp. 465–468

[24] CHENEY E.W.: 'Introduction to approximation theory' (AMS, 2000, 2nd edn.)

[25] MULLER J.-M.: 'Elementary functions: algorithms and implementation' (Birkhäuser, 2006, 2nd edn.)

[26] FRASER W.: 'A survey of methods of computing minimax and near-minimax polynomial approximations for functions of a single independent variable', *J. ACM*, 1965, **12**, (3), pp. 295–314

[27] PLATANIOTIS K.N., VENETSANOPOULOS A.N.: 'Color image processing and applications' (Springer-Verlag, 2000)

[28] KENDER J.R.: 'Saturation, hue, and normalized color: calculation, digitization effects, and use'. Tech. Rep.

Carnegie Mellon University Computer Science Department, 1976

[29] UMBAUGH S.E., MOSS R.H., STOECKER W.V.: 'Automatic color segmentation of images with application to detection of variegated coloring in skin tumors', *IEEE Engng. Med. Biol. Mag.*, 1989, **8**, (4), pp. 43–52

[30] MARDIA K.V., JUPP P.E.: 'Directional statistics' (John Wiley, 2001, 2nd edn.)

doi: 10.1049/iet-ipr.2008.0172