16th European Signal Processing Conference (EUSIPCO 2008), Lausanne, Switzerland, August 25-29, 2008, copyright by EURASIP

PROJECTIVE-SPACE COLOUR FILTERS USING QUATERNION ALGEBRA

Todd A. Ell†, *Stephen J. Sangwine*‡

† 5620 Oak View Court,
Savage, MN 55378, USA
email: T.Ell@IEEE.org

‡ Department of Computing and Electronic Systems,
University of Essex, Wivenhoe Park,
Colchester CO4 3SQ, UK
email: S.Sangwine@IEEE.org

ABSTRACT

Instead of mapping colour image pixels into Euclidean vectors as is conventionally done in colour image processing, we present the idea of using projective space mapping based on homogeneous coordinates. This approach offers a much richer set of geometric operations in the colour space compared to the Euclidean geometry operations that exist in classical colour spaces. The projective geometry of points (pixel values) is described and compared to the classical Euclidean view of pixel values as vectors. The use of homogeneous coordinates is introduced and the geometric operations that are possible are outlined. Then it is shown how colour image pixel values may be transformed into and out of homogeneous coordinates, based on a representation in both cases using quaternions. We then show some examples of colour image operations that offer potential for new types of vector filter and we discuss the possibilities.

1. INTRODUCTION

In colour image processing, colour pixels are traditionally treated as geometric vectors in a Euclidean colour space. Typically, the three RGB colour directions become the basis vectors of the colour space (similarly for other colour spaces such as YC_bC_r). This allows colour image filters to be designed using Euclidean geometry, and many non-linear and a few linear filters have been designed in this way. A significant limitation of Euclidean spaces is the limited number of fundamental geometric operations that are available, such as rotations, translations, dilatations, reflections. Our previous work on vector filtering has led us to the conclusion that we need a greater repertoire of geometric operations and in this paper we present an approach that offers this based on homogeneous coordinates. Whereas in Cartesian coordinates we are limited to Euclidean geometry, in homogeneous coordinates we can use projective geometry and this opens up some new operations that were not previously available. Crucially, these operations are linear in the homogeneous coordinates, offering the possibility of linear implementation (for example in the Fourier transform domain) even though the operation, when related back to the original Cartesian coordinates, is non-linear. This offers the tantalizing possibility of interesting new filters which may be implemented linearly, but which are not confined to the limited set of geometric operations available in Euclidean geometry. It is worth noting that the limited set of Euclidean operations are still available to us in homogeneous coordinates, so they can be combined with the richer set of operations added by the change in coordinate system. As in our previous work, we use quaternions as an algebra that permits us to represent geometric operations as well as vector Fourier transforms and it turns out that the quaternion algebra provides a very natural way to handle homogeneous coordinates.

Section 2 provides a brief introduction to the use of quaternions as a projective space and provides the distinction between points and vectors as different entities. In §3 a practical method of encoding colour images in homogeneous coordinates is given. This is followed in §4 by a filter designed in this new framework. A short discussion and conclusion is given in §5.

2. QUATERNION AND PROJECTIVE SPACE

In this work we use Hamilton's quaternions in hypercomplex form as $q = w + ix + jy + kz$, where $w,x,y,z \in \mathbb{R}$. The hypercomplex operators i, j and k follow the rule

$$ijk = i^2 = j^2 = k^2 = -1.$$

Quaternions can be split into scalar and vector parts, i.e., $q = s + v$, where the scalar part $s = S[q] = w$ and the vector part $v = V[q] = ir + jg + kb$. Therefore the set of Euclidean vectors $\mathcal{V}$ is the set of quaternions which have zero scalar part, i.e., $\mathcal{V} = \{q \in \mathbb{H} \mid S[q] = 0\}$. In shorthand notation $\mathcal{V} = V[\mathbb{H}]$.

One issue with this representation of vectors is that $\mathcal{V}$ is not closed with respect to quaternion multiplication; the product of two vectors can also result in a scalar or a full quaternion, neither of which is a vector. This means that quaternion-based vector filters must be carefully designed to avoid falling outside the vector set.

However, a full quaternion has an alternative interpretation of representing a point in $\mathbb{R}^3$ with an assigned weight. We use the quaternion identity

$$q = s + v = s[1 + p], \quad \text{where } p = v/s.$$

In this form q is used to represent a *point* at the end of the vector p from the origin with *weight*, s. This interpretation was discussed by Joly [1, pp.263–4] in 1905 and MacFarlane [2, p.35] in 1906[1]. Let $\mathcal{P} = \{q \in \mathbb{H} \mid S[q] \neq 0\}$ denote the set of all such weighted-points. Instead of splitting $\mathbb{H}$ into scalar and vector parts, we have created two disjoint subsets: points $\mathcal{P}$ and vectors $\mathcal{V}$, i.e., $\mathbb{H} = \mathcal{P} \cup \mathcal{V}$. In modern terminology, we have interpreted the quaternion space $\mathbb{H}$ as a real projective space $\mathbb{P}^3$.

Under this interpretation, a unit-weight point at the origin is denoted

$$q = 1[1 + i0 + j0 + k0] = 1.$$

The algebra of vectors in $\mathcal{V} \in \mathbb{H}$ and the algebra of (weighted) points $\mathcal{P} \in \mathbb{H}$, both described using quaternion algebra, are different in subtle ways. For example, the direct sum of two vectors, v_1 and v_2, follows the *parallelogram law*, whereas the direct sum of two (weighted) points, p_1 and p_2, follows the *principle of center-of-mass* as shown in figure 1. This can be seen in the following equation

$$\begin{aligned} q_1 + q_2 &= s_1(1 + p_1) + s_2(1 + p_2) \\ &= \underbrace{(s_1 + s_2)}_{weight}[1 + \underbrace{(s_1 p_1 + s_2 p_2)/(s_1 + s_2)}_{center-of-mass}]. \end{aligned}$$

Likewise, scaling a vector $v \in \mathcal{V}$ by $\alpha \in \mathbb{R}$ changes its length but leaves its direction unchanged, whereas scaling a point $p \in \mathcal{P}$ leaves its position unchanged but changes its weight since $\alpha q = \alpha s[1 + p]$.

†Ell is a Visiting Fellow at the University of Essex. This work was supported in part by the U.K. Engineering and Physical Sciences Research Council under Grants GR/S58621/01 and EP/E010334/1.

[1] MacFarlane calls them mass-vectors after Clerk Maxwell. We prefer the term weighted-points so that points are distinguished from vectors, avoiding confusion with the vector-part of a quaternion.

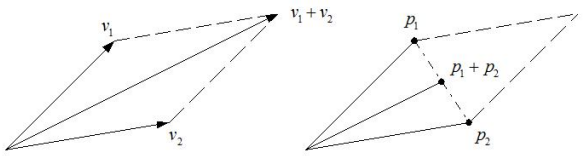


Figure 1: Direct sum of two vectors and two points yield different results.

The set of vectors is closed with respect to quaternion addition and subtraction; however, the set of weighted-points is not. For example, the difference between two equal-weight points, q_1 and q_2, is the *weightless* point (i.e., the vector) that joins them as

$$q_1 - q_2 = s[1+p_1] - s[1+p_2] = 0 + (p_1 - p_2).$$

Table 1 provides a summary of how various quaternion algebraic operations map elements between point and vector subsets.

Quaternion conjugation, denoted with an over-bar, by definition negates the vector part of a quaternion. Consequentially, conjugation and negation of a vector are the same, i.e., $\bar{v} = -v$. In contrast, the conjugate of a weighted-point results in the point's position being reflected across the origin, whereas negation changes the sign of the point's weight, but its position remains the same, i.e., $\overline{s[1+p]} = s[1-p] \neq -s[1+p_2]$.

Not withstanding these differences, some algebraic combinations behave identically for vectors and points. For example, given two vectors, v_1 and v_2, their *linearly weighted* sum (verse direct sum)

$$v_\alpha = (1-\alpha)\, v_1 + (\alpha)\, v_2$$

traces a line from v_1 to v_2 as α varies from 0 to 1, i.e., it draws a line diagonally across the parallelogram defined by v_1 and v_2. Likewise, given two points, p_1 and p_2, their linearly weighted sum

$$p_\alpha = (1-\alpha)\,[1+p_1] + (\alpha)\,[1+p_2] = 1 + [(1-\alpha)\, p_1 + \alpha p_2]$$

also traces the same path across the diagonal of the parallelogram defined by points p_1 and p_2.

Even though points and vectors are disjoint subsets, they share a common framework when viewed as a projective space. For example, a vector can represent a point at infinity along its direction. This can be seen by allowing the weight s, at a point $p = v/s$, to approach zero while keeping the vector-part v fixed so that its position increases without limit. Geometrically, all the points at infinity have been included into a single 3-space model.

The use of homogeneous coordinates is well known in the computer graphics literature where it is used to allow writing of important transformations including: rotations, scaling, skewing, translation, and perspective distortion.

Mapping	Projective Space Interpretation
$\alpha \circ \mathcal{V} \to \mathcal{V}$	*Scales vector length, direction unchanged.*
$\alpha \circ \mathcal{P} \to \mathcal{P}$	*Scales point weight, position unchanged.*
$\mathcal{V} \pm \mathcal{V} \to \mathcal{V}$	*Parallelogram law.*
$\mathcal{P} \pm \mathcal{V} \to \mathcal{P}$	*Translates point in direction of vector.*
$\mathcal{P} + \mathcal{P} \to \mathcal{P}$	*Principle of center-of-mass.*
$\mathcal{P} - \mathcal{P} \to \mathcal{V}$ or $\mathcal{P}$	$\mathcal{V}$ *iff weights are equal.*
$\mathcal{V} \circ \mathcal{V} \to \mathcal{V}$ or $\mathcal{P}$	$\mathcal{V}$ *iff vectors are perpendicular.*
$\mathcal{P} \circ \mathcal{P} \to \mathcal{V}$ or $\mathcal{P}$	$\mathcal{V}$ *iff* $p_1 \cdot p_2 = 1$.

Table 1: Summary of point ($\mathcal{P}$) and vector ($\mathcal{V}$) algebraic mapping operations. Here '$\circ$' denotes quaternion multiplication, '$\cdot$' denotes standard vector dot-product, and $\alpha \in \mathbb{R}$.

Generally data available for analysis is represented in Cartesian not homogeneous coordinates. Hence a method of conversion between coordinate systems is needed. The next section addresses this issue for color image data.

3. COLOUR IMAGE REPRESENTATION

A pixel at image coordinates (n,m) in an RGB image can be encoded as a weighted-point as

$$f_{n,m} = 1 + i\, r_{n,m} + j\, g_{n,m} + k\, b_{n,m}$$

where $r_{n,m}$ is the red component, and $g_{n,m}$ and $b_{n,m}$ are the green and blue components of the pixel respectively. The only change between this and previous Euclidean vector methods [3, 4] is the addition of the unit scalar to each pixel. The choice of a unit scalar (hence a unit weight) is made to simplify the encoding process, otherwise the pixel values would need to be scaled by the chosen weight. The same weight, however, must be used across the entire image, otherwise the decoding process will not return the original image. An image encoded in this way is referred to as being in *weighted-point* form.

Decoding an image pixel in weighted-point form into colour pixel values involves simply scaling the vector-part by the scalar-part to determine its colour-space position as

$$p_{n,m} = V[f_{n,m}] / S[f_{n,m}]$$

This is a point-wise operation. In this process the weight of each point is discarded. This is not an issue since the weights were arbitrarily assigned during the encoding of the image.

The authors [5] have already noted the advantage of placing the origin of the RGB colour-space at the center of the colour cube. These include: all pixels with a common direction from mid-grey (the origin) have the same hue; a pixel vector represents by its length and direction how much the pixel differs from mid-grey; and finally, all directions from the origin represent a colour. All these advantages carry over to pixel points. Therefore images in this work use a mid-grey origin for the colour space.

4. ILLUSTRATIVE EXAMPLES

In this section we provide concrete examples of filtering operations applied to colour image processing. But first we start with the basic projective algebra as applied to whole images to highlight what appears to be trivial aspects of the shift from Cartesian to homogeneous coordinates, when in fact they are not trivial in their effects.

4.1 Sum and Product of two images

All filtering operations consist of algebraic combinations (Euclidean and now projective) of image pixel values. Ideally these combinations would be related to perceptual properties of colour; or at the least they should not induce harsh perceptual distortions. In Cartesian coordinates the direct addition of two pixels follows the parallelogram law, so it is common for two random coloured pixels that their sum will fall outside the colour cube and needs to be clipped to yield a valid colour. Since in homogeneous coordinates the result is governed by the principle of center-of-mass the addition of *any* two pixel values will *always* fall within the colour cube. This is true since the colour cube is a closed convex shape, hence the line between any two points is entirely contained in the colour cube. Figure 2 shows two images (tiffany and sailboat) that were summed and multiplied in both Cartesian and homogeneous coordinates. As expected the Cartesian sum shows saturated values due to the vector sum falling outside the colour cube. A valid colour is shown after application of the necessary clipping algorithm [5] to return them to the colour cube. In contrast the homogeneous sum induces no such behavior. The Cartesian and homogeneous products show the same situation; whereas the Cartesian product is all

(a) Image f_1 (b) $q_1 + q_2$ (c) $q_1 \circ q_2$

(d) Image f_2 (e) $v_1 + v_2$ (f) $v_1 \circ v_2$

Figure 2: Pixel-wise sum & product of two images (a) & (d) using both Euclidean vector v_i and Projective point $q_i = s_i[1 + p_i]$ geometry.

but unrecognizable, the homogeneous product clearly shows both images[2].

4.2 A Projective-Space Colour Filter

The signal block diagram shown in figure 3 will be used to illustrate some of the colour filtering features possible in projective space. The filter design is split into three stages: a colour-discrimination stage D, a spatial filtering stage F, and a colour-response stage R.

The colour discriminator D is designed to isolate a specific colour feature within the input image. For example it can be used to detect colour-specific edges such as red-to-green transitions, or the proportion of the colour in parallel to a specific colour direction. The output of the discriminator is a scalar image, i.e., $D : \mathbb{H}^{n\times m} \to \mathbb{R}^{n\times m}$. The spatial filter stage F applies standard grey-scale filtering to the discriminator output image $d_{n,m}$ and yields a blending factor $k_{n,m}$. It can, for example, be used to smooth or sharpen the discriminator output. For reasons to be shown later, the output of this stage is normalized to the ± 1 range. Finally, the colour response stage R uses two images and the principal of center-of-mass to blend these two images proportionally to the value of $k_{n,m}$. One of the images is the original input image, the companion image defines which *directions* in colour space the blending should move the original image. For example, if the companion image is entirely white, then the blending factor $k_{n,m}$ determines the input pixel's shift toward white. But, if the companion image is entirely black, then $k_{n,m}$ determines the input pixels shift toward black. However, if the companion image is multi-coloured, then each pixel in the input image is shifted toward its companion pixel.

Figure 4 shows the result of one such projective space colour filter on the familiar Lena image. The discrimination stage for this example is designed to identify yellow regions of the image. An equation which does this is

$$d_{n,m} = \tfrac{1}{\sqrt{2}} \left| f_{n,m} - \mu f_{n,m} \mu - 2 \right|$$

where $\mu \in V[\mathbb{H}]$, $|\mu| = 1$ and points to a colour-of-interest (COI). What this equation does is project each pixel position in colour space (as a vector) onto the COI, hence it represents the relative strength of the COI-component of that pixel. The image of figure 4(c) shows the discriminator output when μ is set using $(R,G,B) = (237,204,164)/255$ which is from the yellow band in the lower right-hand corner of the Lena image. As can be seen in this figure, the white areas correspond to the yellow regions of the image. To further simplify this example, no spatial filtering is applied; hence the blending factor $k_{n,m} = d_{n,m}$. The final stage is where the design comes together. The goal is to have all non-yellow pixels pushed toward mid-grey but leave the yellowish pixels alone. An equation which does exactly this is

$$g_{n,m} = (1 + k_{n,m})\, f_{n,m} + (1 - k_{n,m})\, \bar{f}_{n,m},$$

which is the linear sum of the input image $f_{n,m}$ and its companion image $\bar{f}_{n,m}$. As noted at the end of section 2, conjugating a

[2]We acknowledge discussions with David Alleysson in Grenoble during 2005 in which the idea of colour image products was first introduced to us.

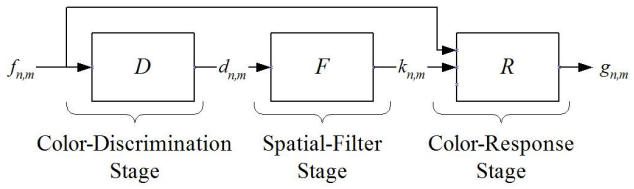


Figure 3: 3-Stage Colour-Sensitive Filter Block Diagram.

weighted-point reflects the point across the origin. Since the encoding process places the origin at mid-grey, this means that $\bar{f}_{n,m}$ is the opponent colour of the original image $f_{n,m}$. The opponent colour image of Lena is shown in figure 4(b). Now, since $k_{n,m} \in [0,+1]$ this means the output image moves each pixel which corresponds to low yellow content (i.e., $k \to 0$) toward its opponent colour, along a fixed hue due to the center-of-mass property of the projective space. But, for pixel values with high yellow content (i.e., $k \to 1$), the pixel remains closer to the original value. The results of this process are shown in figure 4(d). All non-yellow portions are driven toward mid-grey, but the yellow portions are unchanged.

(a) Input, $f_{n,m}$ (b) $\bar{f}_{n,m}$

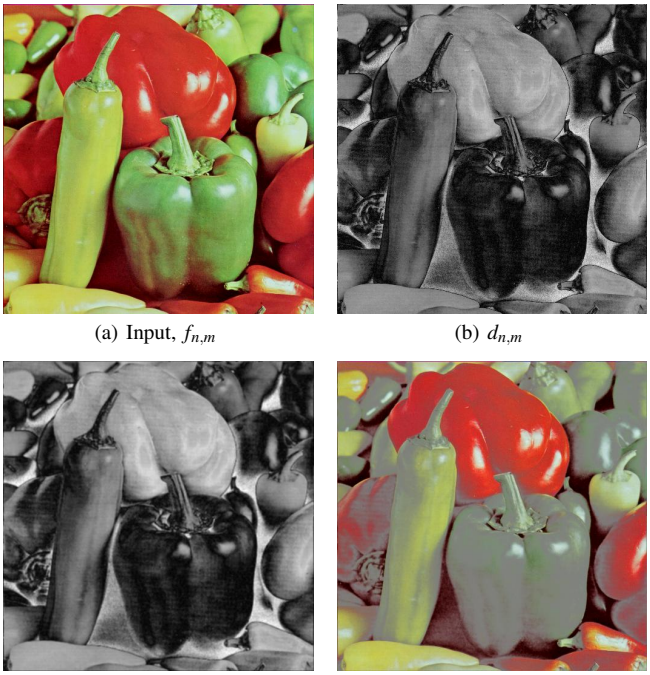
(c) $d_{n,m} = k_{n,m}$ (d) Output, $g_{n,m}$

Figure 4: Projective-space Colour Filter Example #1. This filter is designed to highlight yellow portions of the input image by driving all non-yellow pixels toward mid-grey.

Figure 5 shows the same filter applied to the standard *peppers* image. Here the filter is used to isolate the red pixels (i.e., the COI μ is set using $(R,G,B) = (1,0,0)$). This time a 3×3 pixel averaging filter is used to smooth the discriminator output.

5. DISCUSSION AND CONCLUSION

In this paper we have presented the use of a quaternion based projective space to represent colour images which gives an alternative to their representation as Euclidean vectors in RGB colour space. In projective space, pixel values are represented as weighted-points in homogeneous coordinates. Familiar equations in homogeneous co-



(a) Input, $f_{n,m}$ (b) $d_{n,m}$

(c) $k_{n,m}$ (d) Output, $g_{n,m}$

Figure 5: Projective Colour Filter Example #2. This filter is designed to highlight red portions of the image by driving non-red pixels to mid-grey. An eight-neighbor averaging filter is applied to the discriminator output.

ordinates behave subtly different in Cartesian coordinates but these differences can be exploited. For example, it was shown that the basic operations of addition and multiplication of weighted-point pixels do not have the same perceptual distortions as their Euclidean vector counterparts.

Also, a non-linear filter was presented to illustrate further these differences. This simple filter separated input colour discrimination from output colour response as two distinct stages with the spatial filtering operations handled by an intermediate grey-scaler filter between them. This was possible in homogeneous coordinates due to the principle of center-of-mass and the fact that scaling a singleton pixel in homogeneous coordinates does not change its colour; only the relative scaling of two or more pixels alters the resulting colour.

This new representation of pixel values as weighted-points in homogeneous coordinates extends the repertoire of *linear* geometric operations available for image filtering beyond rotations, dilations, and reflections of pixel values in colour-space. It is known, through fundamental geometric concepts, that linear operations in homogeneous coordinates include translations and perspective transforms; neither of which is a linear operation in Cartesian coordinates.

References

[1] Charles J. Joly. *A manual of quaternions*. Macmillan, London, 1905. Available online at Cornell University Library: `http://historical.library.cornell.edu/math/`.

[2] Alexander MacFarlane. *Vector analysis and quaternions*. John Wiley and Sons, New York, 4th edition, 1906. Available online at Cornell University Library: `http://historical.library.cornell.edu/math/`.

[3] S. J. Sangwine and T. A. Ell. Colour image filters based on hypercomplex convolution. *IEE Proceedings – Vision, Image and Signal Processing*, 147(2):89–93, April 2000.

[4] C. E. Moxey, S. J. Sangwine, and T. A. Ell. Vector phase correlation. *Electron. Lett.*, 37(25):1513–5, 6 Dec 2001.

[5] S. J. Sangwine and T. A. Ell. Gray-centered RGB color space. In *Second European Conference on Color in Graphics, Imaging and Vision (CGIV 2004)*, pages 183–186, Technology Center AGIT, Aachen, Germany, 5–8 April 2004. The Society for Imaging Science and Technology.