Available online at www.sciencedirect.com

ScienceDirect

Computer Vision and Image Understanding 107 (2007) 88–96

Computer Vision and Image Understanding

www.elsevier.com/locate/cviu

# Quaternion color texture segmentation

Lilong Shi, Brian Funt *

*School of Computing Science, Simon Fraser University, 8888 University Drive, Burnaby, BC, Canada V5A 1S6*

Received 4 May 2006; accepted 21 November 2006
Available online 26 January 2007
Communicated by Rastislav Lukac

**Abstract**

The quaternion representation of color is shown here to be effective in the context of segmenting color images into regions of similar color texture. The advantage of using quaternion arithmetic is that a color can be represented and analyzed as a single entity. A low-dimensional basis for the color textures found in a given image is derived via quaternion principal component analysis (QPCA) of a training set of color texture samples. A color texture sample is then projected onto this basis to obtain a concise (single quaternion) description of the texture. To handle the large amount of training data, QPCA is extended to incremental QPCA. The power of the proposed quaternion color texture representation is demonstrated by its use in an unsupervised segmentation algorithm that successfully divides an image into regions on basis of texture.
© 2007 Elsevier Inc. All rights reserved.

*Keywords:* Quaternion color processing; Quaternion principal components analysis; Color texture analysis; Image segmentation; Color texture classification

## 1. Introduction

Many different local image features have been proposed for characterizing textures [1–3,8]. Since there are many possible texture features, finding a feature vector with good discriminating power is important. For color imagery, Hoang et al. [8] showed that using color and texture in combination results in better discrimination than using the color and texture features separately.

We propose a new color texture classification method based on the quaternion representation of a color image. Quaternions provide a new way to process color and texture in combination. The advantage of the quaternion representation of color, as proposed by Sangwine [4,5,7], is that it combines a color 3-tuple (RGB or LMS) into a single hypercomplex number. A color can then be processed as a unit, rather than as 3 separate channels. A quaternion has a real part and three imaginary parts and can be written as $q = a + b \cdot i + c \cdot j + d \cdot k$. An RGB color triple is represented as a purely imaginary quaternion of the form $R \cdot i + G \cdot j + B \cdot k$. Both the Fourier transform [5] and principal component analysis [6,7] have been extended to quaternion arithmetic.

Standard principal component analysis (PCA) has been used previously in the texture context, but our use here is different. Previously PCA was used to reduce the dimensionality of the features extracted from grayscale textures [8,10]; whereas, in our case, quaternion principal component analysis (QPCA) is used to reduce the dimensionality of the raw textures directly. In particular, we use QPCA with colors encoded as quaternions to calculate a quaternion basis for color textures. The basic idea is to use QPCA to obtain a low-dimensional representation of textures sampled from image subwindows. QPCA computes a basis ordered in terms of the amount of variance accounted for in the data. The low-dimensional approximation is constructed by projecting the original input onto the first few basis vectors. Two textures are compared by projecting each onto the first few basis vectors and then comparing the resulting coefficients.

* Corresponding author.
*E-mail address:* funt@sfu.ca (B. Funt).

To test whether or not the QPCA representation of color texture is effective, we use it to segment images into regions of homogeneous texture and make a qualitative comparison to the previously published results [8,11]. Quaternion-based color texture segmentation proceeds in three main stages: feature extraction, feature classification, and region merging. Fig. 1 gives an overview. In the feature-extraction stage, feature vectors are generated by applying QPCA to training data obtained from a set of subwindows taken from the input image. The subwindows are samples of the image's textures and should be large enough to characterize a texture. Each $w$-by-$w$ subwindow is re-represented as a vector of $w^2$ quaternions, one quaternion per pixel. Applying QPCA to this collection of vectors yields a quaternion basis for the image's textures. As with PCA, the QPCA basis is ordered in terms of the fraction of the variance accounted for by each basis vector. The original $w^2$ dimensionality can be reduced by projecting each subwindow onto only the first few basis vectors. We use these quaternion basis vectors as a texture–feature space. It has the advantage that because of the quaternion color representation, it characterizes both the spatial and color aspects of the image's textures. In the classification stage, texture features are grouped via k-means clustering and the input image pixels are labeled according to this clustering. Finally, in the segmentation stage which incorporates spatial information, regions of similar texture are merged to produce the final image segmentation.

## 2. Quaternions and quaternion filtering

A model of color texture needs to take into account not only the spatial interaction within each of the three color channels, but also the interaction between different color channels. Panjwani and Healey [11] modeled these spatial and inter-channel interactions explicitly using a Gaussian Markov Random Field Model in which for each pixel the expected value for each channel is derived from a probabilistically weighted linear combination of the RGB triplets at neighbouring pixels. The quaternion representation of RGB color texture also represents spatial intra-channel and inter-channel relationships, and allows for both linear and non-linear interactions.

When multiplying two quaternions, each of their components interacts with all other components, and the effect spreads over all components. Given two quaternions $a_q = a_0 + a_1 \cdot i + a_2 \cdot j + a_3 \cdot k$ and $x_q = x_0 + x_1 \cdot i + x_2 \cdot j + x_3 \cdot k$ with real parts denoted as $R[a_q]$ and $R[x_q]$ and the three imaginary parts denoted as $I[a_q]$ and $I[x_q]$. The product of $a_q$ and $x_q$ is

$$y_q = a_q \times x_q = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \times \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} a_0x_0 - a_1x_1 - a_2x_2 - a_3x_3 \\ a_1x_0 + a_0x_1 - a_3x_2 + a_2x_3 \\ a_2x_0 + a_3x_1 + a_0x_2 - a_1x_3 \\ a_3x_0 - a_2x_1 + a_1x_2 + a_0x_3 \end{bmatrix}, \quad (1)$$

where $\times$ denotes quaternion multiplication. If $a_0 = 0$ and $x_0 = 0$, so that $a_q$ and $x_q$ are pure quaternions, then the product $y_q = a_q \times x_q$ can be decomposed as

$$R[y_q] = a_1x_1 + a_2x_2 + a_3x_3 = -dot(I[a_q], I[x_q]) \quad (2)$$

$$I[y_q] = cross(I[a_q], I[x_q]). \quad (3)$$

When the imaginary components of $a_q$ and $x_q$ represent two RGB triples, then the multiplications involved when QPCA computes the correlation matrix incorporate

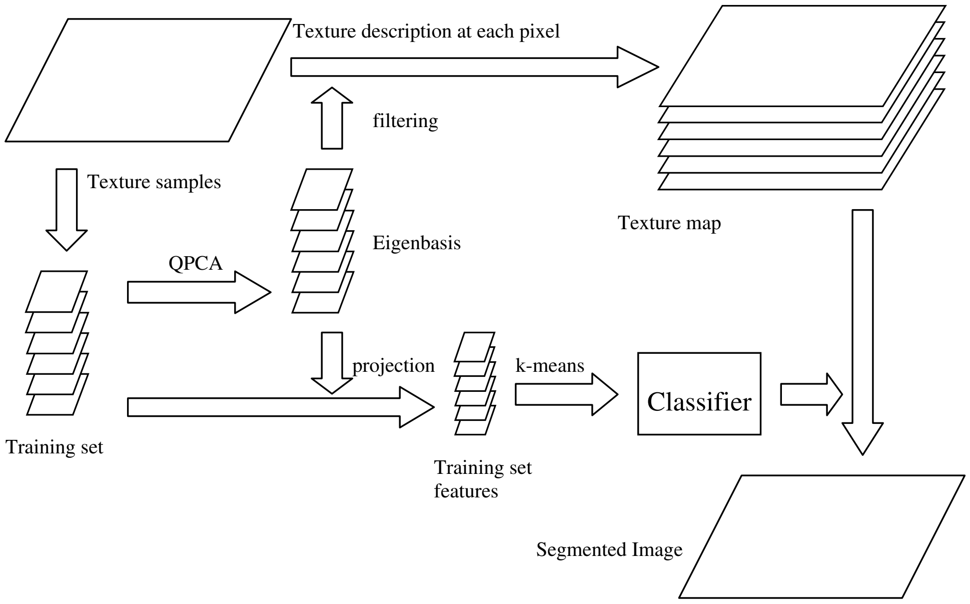

Fig. 1. Overview of the quaternion texture segmentation process. QPCA is applied to texture samples from image subwindows. The input image is then filtered by the resulting basis vectors producing a texture map. A classifier derived from a k-means clustering of the training set data labels each pixel according to the texture map, and then regions of similar texture are merged. See text for further details.

interactions between the different color channels. For instance, the cross product of two quaternion colors creates a color perpendicular to both.

Quaternion-valued filters have been introduced and used for color-edge detection and smoothing by Sangwine [4]. In addition, quaternion Fourier coefficients [5] can be viewed as sine/cosine quaternion filter responses to quaternion vectors. Quaternion filters are usually designed such that each entry is a full quaternion number, but the input data are pure quaternions. In this paper, the input image is filtered to produce and output a texture–feature map. The texture filters are obtained by QPCA, and then applied to the image by convolution.

In order to study the filter responses of full quaternion filters over a pure quaternion data space, consider the result of multiplying a full quaternion with a pure quaternion. When $x_0 = 0$, $x_q$ is a pure quaternion, and the multiplication $y_q$ of $a_q$ and $x_q$ is:

$$R[y_q] = a_1x_1 + a_2x_2 + a_3x_3 = -\mathrm{dot}(I[a_q], I[x_q]) \quad (4)$$

$$I[y_q] = R[a_q]I[x_q] + \mathrm{cross}(I[a_q], I[x_q]) \quad (5)$$

Suppose $F$ is a full quaternion filter, and $Q$ is a pure quaternion color image, then the filter response obtained by convolving $F$ with $Q$ is:

$$\begin{aligned}\mathrm{Conv}(F,Q) &= -\mathrm{conv}_1(F,Q) + \mathrm{conv}_2(F,Q)\\ &\quad + \mathrm{conv}_3(F,Q), \quad (6)\end{aligned}$$

$$\mathrm{conv}_1(F,Q) = (F_r \otimes Q_r + F_g \otimes Q_g + F_b \otimes Q_b, 0, 0, 0) \quad (7)$$

$$\mathrm{conv}_2(F,Q) = (0, F_l \otimes Q_r, F_l \otimes Q_g, F_l \otimes Q_b) \quad (8)$$

$$\mathrm{conv}_3(F,Q) = (0, F_{r,g,b} \oplus Q_{r,g,b}), \quad (9)$$

The operator $\otimes$ denotes the normal convolution of two real vectors. The subscripts $l$, $r$, $g$, $b$ indicate the real and three imaginary parts. The operator $\oplus$ denotes a special convolution with the cross product as multiplication of each pixel pair. The resulting convolution of quaternion vector **F** and quaternion vector **Q** is also a quaternion vector, which contains four layers: the $l$ layer is somewhat related to intensity; the $r$ layer corresponds to the red channel; the $g$ layer corresponds to the green channel; and the $b$ layer corresponds to the blue channel. Eq. (7) shows how the $l$ layer is computed in conv$_1$. In Eqs. (8) and (9), the $r$,$g$ and $b$ layers are computed at once. The combination of conv$_1$, conv$_2$ and conv$_3$ in Eq. (6) yields the final convolution result. The entire convolution process can be decomposed as illustrated in Fig. 2.

## 3. Quaternion texture segmentation method

In this section, we detail the algorithm for segmenting an image into regions of homogenous color texture. The first stage is to extract an orthogonal basis for the color textures in an image. This basis is calculated by sampling square windows from the image and expressing the contents of each such window as a vector of quaternions and arranging these vectors as the columns of a matrix. QPCA applied to this matrix yields an orthogonal basis for the contents of the windows ordered in terms of the variance accounted for by each basis vector. Similar to the standard PCA, the dimensionality of the feature space can then be reduced by selecting only the first few bases that account for the majority of the variance. These basis vectors are vectors of quaternions. The contents of any image window can then be approximated concisely by projecting them onto the selected basis. The projection can then be used as the extracted color texture feature of the window. When we consider the entire set of subwindows and the projection of each one onto the quater-

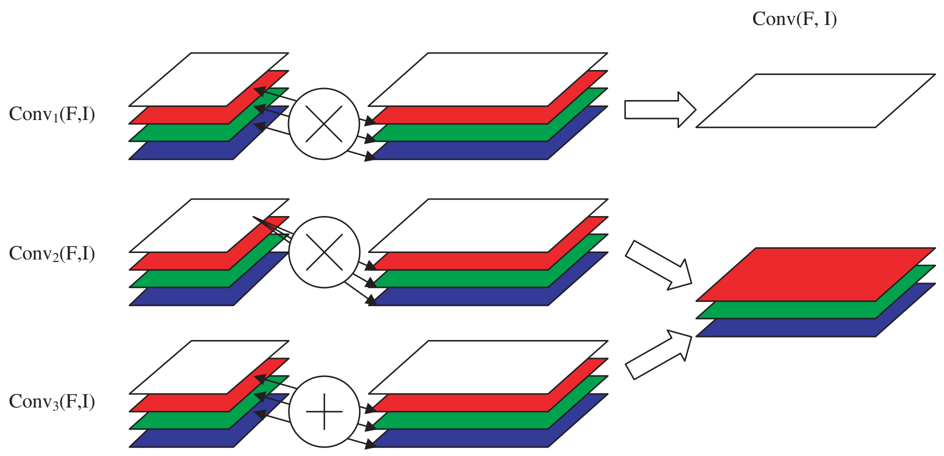

Fig. 2. Diagrammatic overview of a full quaternion filer applied to a pure quaternion image. In Conv$_1$, the RGB layers of the filter convolve with the RGB layers of the image correspondingly and separately. The negated sum of the resulting three filter responses becomes the real component of the final quaternion result. In Conv$_2$, the real layer of the filter convolves separately with each of the R, G and B channels. In Conv$_3$, the RGB channels of the filter "convolve" with the RGB channels of the image. Each multiplication during the convolution is of two RGB 3-tuples—one from the filter, the other from a pixel of the image—and results in their cross product, which is also a pure quaternion. The sum of the result of Conv$_3$ with the result of Conv$_2$ yields the RGB (the imaginary part) of the final quaternion output.

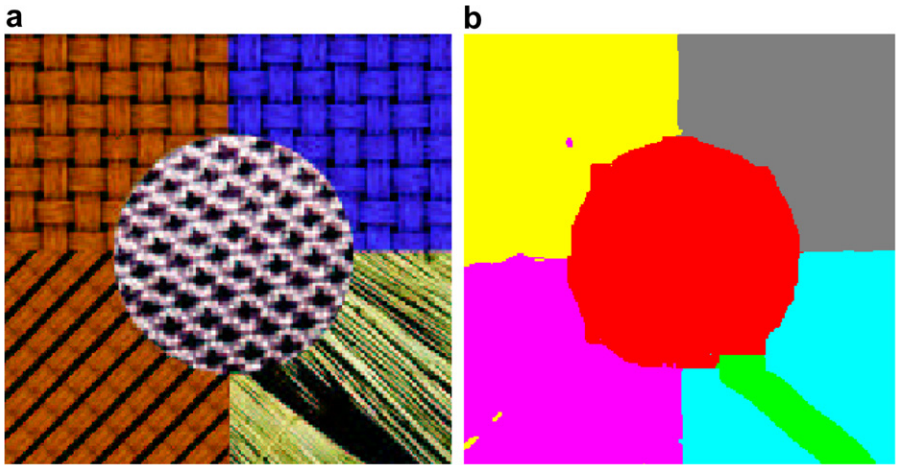

Fig. 3. Texture segmentation results: (a) synthetic input image from Hoang [8] having brown regions on the top left and bottom left, a blue region in the top right, a pale green region in the bottom right and a grayish circular region in the center; (b) segmentation result for (a) showing that the QPCA method successfully separates the top left and top right regions, which differ in color but have similar (although rotated) grayscale structure, and simultaneously separates the top-left and bottom-left regions, which differ in grayscale structure but have similar color. The shadow in the lower right region is identified. The result in (b) is similar to that in [8] (page 272, Fig. 3 (d)) without shadow invariance. Parameters used were: image size $192 \times 192$, window size $13 \times 13$, abutting non-overlapping windows, initial number of k-means clusters 15, similarity threshold 8.5, Gaussian smoothing $\sigma = 2$. A qualitatively similar result was obtained for overlapping windows. Reprinted from *Signal Processing*, Vol (85), M. Hoang, J. Geusebroek, A. Smeulders, Color texture measurement and segmentation, 265–275.

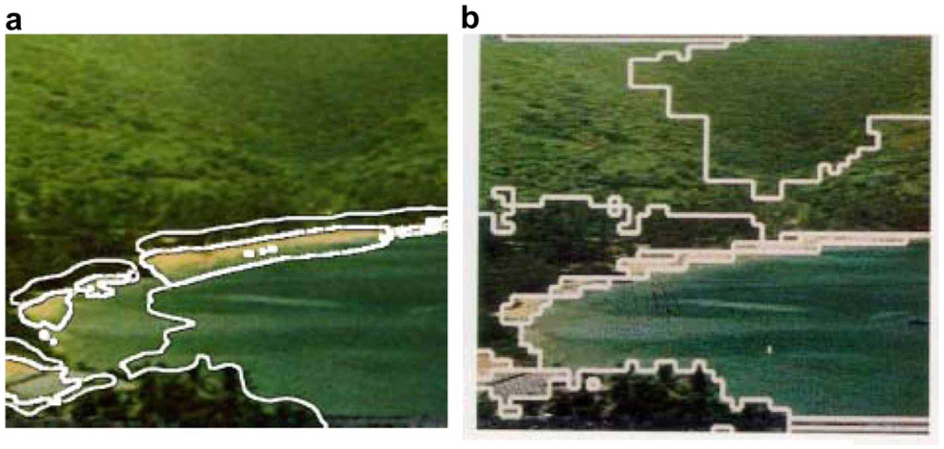

Fig. 4. Comparison of quaternion texture segmentation (a) to Markov random field segmentation [11]. (b) Some discrepancies may arise as because the result in (a) is based on a scan of the published input figure, rather than the original data file.

nion basis, we can view the process as a filtering step in which the image is filtered by the basis.

For image texture segmentation, we are concerned with representing the textures that occur in the given image, not textures in general. Therefore, in the first step the training set of texture samples is drawn from image by using subwindows of size $w \times w$, where $w$ depends on the image resolution. It is chosen to be large enough to cover a representative texture element and small enough so that it will not generally cover multiple texture elements. In order to reduce the computational cost, we do not use subwindows centered on each and every pixel. Instead, we tried a sampling from subwindows centered on random pixels, from overlapping windows and from non-overlapping windows. To further reduce the storage and computational costs, we also developed an Incremental QPCA method as described below that approximates QPCA, but computes the result incrementally.

The RGB pixel values from each window are represented as quaternions and formed into a column vector of quaternions $v_q$ of size $w^2$. The training set of textures is then represented as a matrix $T_q$ with columns $v_q$. QPCA decomposes $T_q$ into singular values and their corresponding quaternion eigenvectors $U_q$. Taking only the first $d$ of the $w^2$ eigenvectors reduces the dimensionality of the texture model. This of course reduces the accuracy with which the training set can be represented, but the expectation is that only features that are irrelevant to texture discrimination

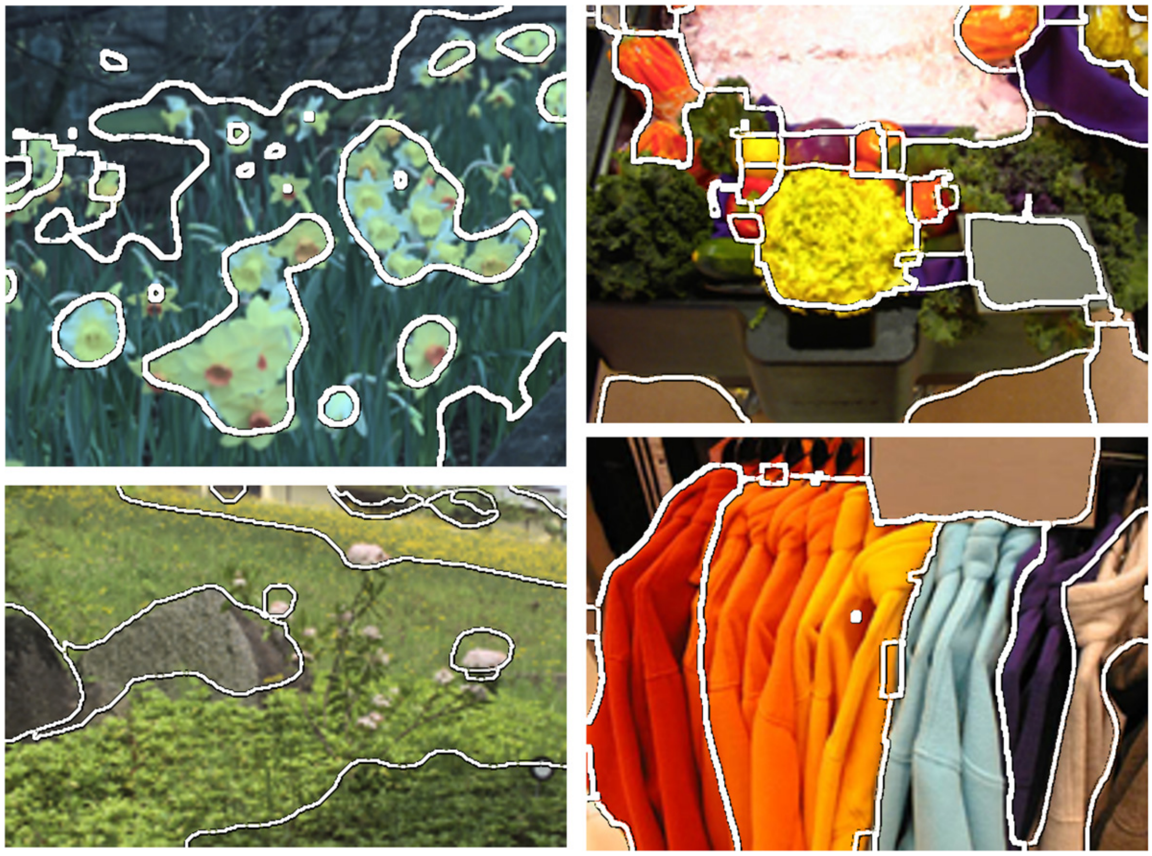

Fig. 5. Segmentation results. Parameters: image size $\approx 200 \times 200$ (original images are resized so that the total number of pixels is approximately 40,000), window size $13 \times 13$, abutting windows, initial number of k-means clusters 15, similarity threshold 8.5, Gaussian smoothing $\sigma = 2$.

will be lost. Surprisingly, we found experimentally that the best choice was $d = 1$, meaning that the method relies on the first basis vector of $U_q$ only. Therefore, a window's quaternion vector $v_q$ of size $w^2$ is reduced to size 1, so its texture is describe by a single quaternion. The resulting quaternion needs no longer be a purely imaginary quaternion and generally does include a non-zero real part, so effectively texture is being represented by a 4-tuple.

The second stage of the quaternion color texture analysis is texture clustering based on the features from each sub-window. With $d = 1$, the whole training set $T_q$ is represented by $T_q^1 = U_q^1 \cdot T_q$, where $U_q^d$ represents the first $d$ eigenvectors. Texture classification begins by k-means clustering of the textures in the training set according to their associated reduced feature vectors $T_q^1$. In other words, the quaternion feature vectors (the columns of $T_q^1$) are input to the k-means algorithm. The output of the $k$-means clustering is $k$ centroids describing the mean features of the texture clusters. The parameter $k$ is chosen to be larger than the number of regions expected in the final region segmentation. For example, $k = 15$ is a good choice. Subsequent region merging will eliminate any extra clusters.

The next step is to classify each image pixel in terms of the texture of $w \times w$ window centered on it. The window's contents are represented as a vector of quaternions and then projected onto the basis $U_q^1$ producing feature vector $v_q^1$ (a single quaternion). The pixel feature vectors are then smoothed spatially by Gaussian smoothing with standard deviation $\sigma$. The pixel's texture is assigned a label based on the training set cluster to which it is closest in terms of its Euclidean distance to the cluster's centroid.

The final stage in the texture segmentation process is region merging. The merging proceeds iteratively and is based on combining statistically similar regions. At each iteration, the two most similar clusters are merged according to a region-similarity measure. The similarity of two regions $R_i$,$R_j$ with mean feature vectors $\mu_i$ and $\mu_j$ and covariance matrices $\Phi_i$ and $\Phi_j$ is defined by Nguyen et al. [9] as:

$$S_{i,j} = (\mu_i - \mu_j)^T [\Phi_i + \Phi_j]^{-1} (\mu_i - \mu_j) \quad (10)$$

The smaller $S_{i,j}$, the more similar the clusters. At each iteration, the two regions with the smallest $S_{i,j}$ are merged until no $S_{i,j}$ is less than a specified threshold. Finally, due to the fact that the sample window may cover more than one region, a post-processing step is needed to eliminate spurious segmentations at region boundaries. If a small neighbourhood region around a pixel includes three or more different texture labels then the pixel in the middle is assigned to the nearest of the other two regions. Fig. 1 summarizes the entire segmentation process.

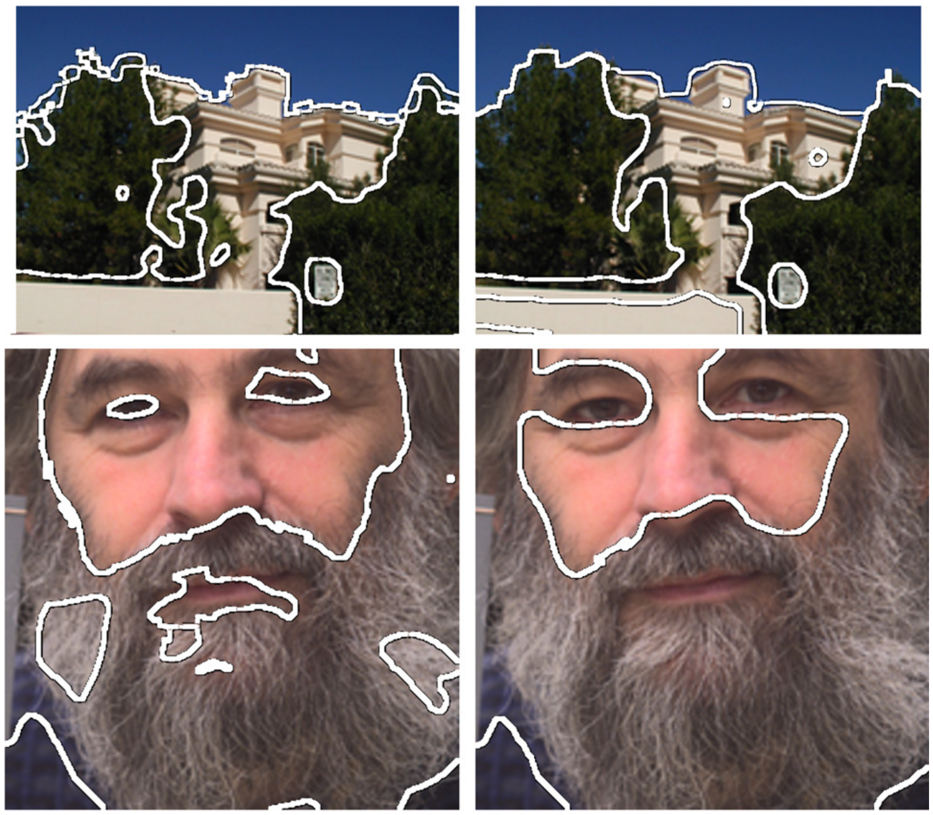

Fig. 6. The effect of subwindow size on the segmentation results. Parameters: image size $\approx 200 \times 200$, abutting windows, initial number of k-means clusters 15, Gaussian smoothing $\sigma = 2$. The window size $7 \times 7$ for the left column and $13 \times 13$ for the right column. The segmentations based on smaller subwindows tend to capture more localized texture features.

## 4. Incremental quaternion principle component analysis

The implementation of QPCA is based on QSVD (quaternion singular value decomposition), which can be considered a generalization of real or complex number singular value decomposition and inherits similar properties. Applications of QSVD to color image compression have been proposed by Sangwine [7] and Pei [6]. The eigenvalues and eigenvectors of a quaternion matrix are also discussed by them. Two important observations are that only the right eigenvalues are defined, and that each quaternion matrix has an equivalent complex matrix. From this latter point it follows that existing complex SVD algorithms can be applied to the equivalent complex matrix to obtain the eigenvectors and singular values of the corresponding quaternion matrix.

The algorithm for QSVD is as follows [7]:

1. Given a zero-centered quaternion-valued matrix $\mathbf{X}_q$.
2. Calculate the equivalent complex matrix $\mathbf{X}_c$ based on $\mathbf{X}_q$.
3. Apply complex SVD on $\mathbf{X}_c$ to get the complex basis $\mathbf{U}_c$ and singular values $\mathbf{S}_c$.
4. Calculate the quaternion basis $\mathbf{U}_q$ and singular values $\mathbf{S}_q$ of $\mathbf{X}_q$ based on $\mathbf{U}_c$ and $\mathbf{S}_c$.

SVD for large data sets can be computationally expensive, or even infeasible, and hence so is the quaternion SVD. For that reason, we extend SVD to an incremental QSVD (IQSVD) algorithm that is analogous to the ISVD algorithm for real numbers [12,13]. Given a new data point to add to an existing data set, IQSVD incrementally updates its existing QSVD model of the data set. Suppose a data set of $N$ training samples $x_i \in Q^n (i = 1, \ldots, N)$ in quaternion space has been used to derive a quaternion eigenspace model $\Omega = (\bar{x}, U, \Lambda, N)$ composed of eigenvectors and eigenvalues, where $\bar{x}$ is the mean of input quaternion data vectors, U is a $n \times r$ quaternion matrix whose column vectors correspond to the quaternion eigenvectors, and $\Lambda$ is a $k \times k$ matrix the diagonal elements of which correspond to the eigenvalues, and k is the dimensionality of the subspace. If then another training sample $y$ is added to the data set, both the mean vector and covariance matrix change, and the eigenvectors and eigenvalues, therefore, should also be recalculated. Updating the mean input quaternion vector $\bar{x}$ is straightforward:

$$\bar{x}' = \frac{1}{N+1}(N\bar{x} + y) \tag{11}$$

Updating the quaternion eigenvectors and eigenvalues is more complicated. First, we need to compute the residual vector $h$ that tells how accurately the existing eigenmodel approximates $y$:

$$h = (y - \bar{x}) - U \cdot g, \text{ where } g = U^T (y - \bar{x}) \tag{12}$$

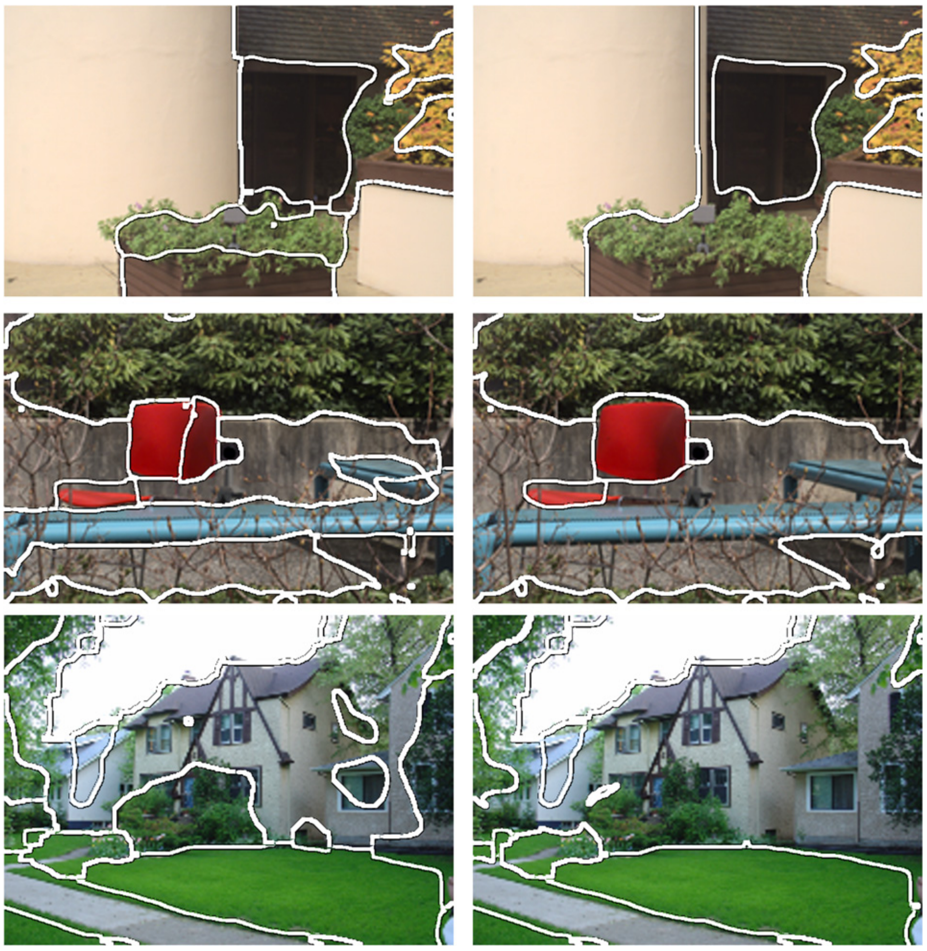

Fig. 7. The effect of the similarity measure on the segmentation results. Parameters: image size $\approx 200 \times 200$, window size $13 \times 13$, abutting windows, initial number of k-means clusters 15, Gaussian smoothing $\sigma = 2$. The similarity threshold is higher for the two images on the right column, the similarity threshold is lower for the two images on the left column.

For real and complex numbers, it has been shown that the eigenvectors and eigenvalues can be updated based on the solution of an intermediate problem which is to form a matrix $D$ that substitutes for the original the data set [12]. First solve $D$ for $R$

$$D = \frac{N}{N+1}\begin{bmatrix} \mathbf{\Lambda} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} + \frac{N}{(N+1)^2}\begin{bmatrix} \mathbf{g}\mathbf{g}^{\mathrm{T}} & \gamma\mathbf{g} \\ \gamma\mathbf{g}^{\mathrm{T}} & \gamma^2 \end{bmatrix} = R\Lambda' R^{\mathrm{T}} \tag{13}$$

where $\gamma = \hat{h}^T(y - \bar{x})$, $R$ is an $(r+1)\times(r+1)$ matrix whose column vectors correspond to the eigenvectors obtained from $D$, $\Lambda'$ is the new eigenvalue matrix, and $\mathbf{0}$ is a r-dimensional zero vector. Using the solution $R$, the new $n \times (r+1)$ eigenvector matrix $U'$ can be computed as follows [12]:

$$U' = [U, \hat{h}]R \tag{14}$$

where $\hat{h} = \begin{cases} h/\|h\| & \text{if} \|h\| > 0 \\ 0 & \text{otherwise} \end{cases}$

The matrix $R$ rotates the combination of the old eigenvectors and the residue of the new data point to span a new subspace in which each eigenvector is still orthogonal. In the quaternion case, the derivation and proof are exactly the same. The only difference is that we replace all real operations with quaternion operations.

An algorithm for incremental QPCA (IQPCA) based on the QSVD method above is as follows:

1. Given a quaternion-valued matrix $\mathbf{X}_q$.
2. Compute mean and zero-center $\mathbf{X}_q$.
3. Calculate quaternion basis $\mathbf{U}_q$ and singular values $\mathbf{S}_q$ of $\mathbf{X}_q$ by QPCA.
4. For a new input quaternion vector $\mathbf{v}_q$.
   (a) Update the mean.
   (b) Solve the intermediate problem with matrix $\mathbf{D}_q$.
   (c) Calculate the new basis $\mathbf{U}_q$ and new singular values $\mathbf{S}_q$.
   (d) Delete the eigenvector associated with the smallest singular value to maintain an eigenbasis of dimension $r$.

This makes no allowance for a change in the number of basis vectors; however, it is easy to add a threshold ε above which the number of basis vectors is expanded to achieve a specific level of accuracy. In the texture case, only the very first few eigenvectors are used in any case, so $r$ can be set

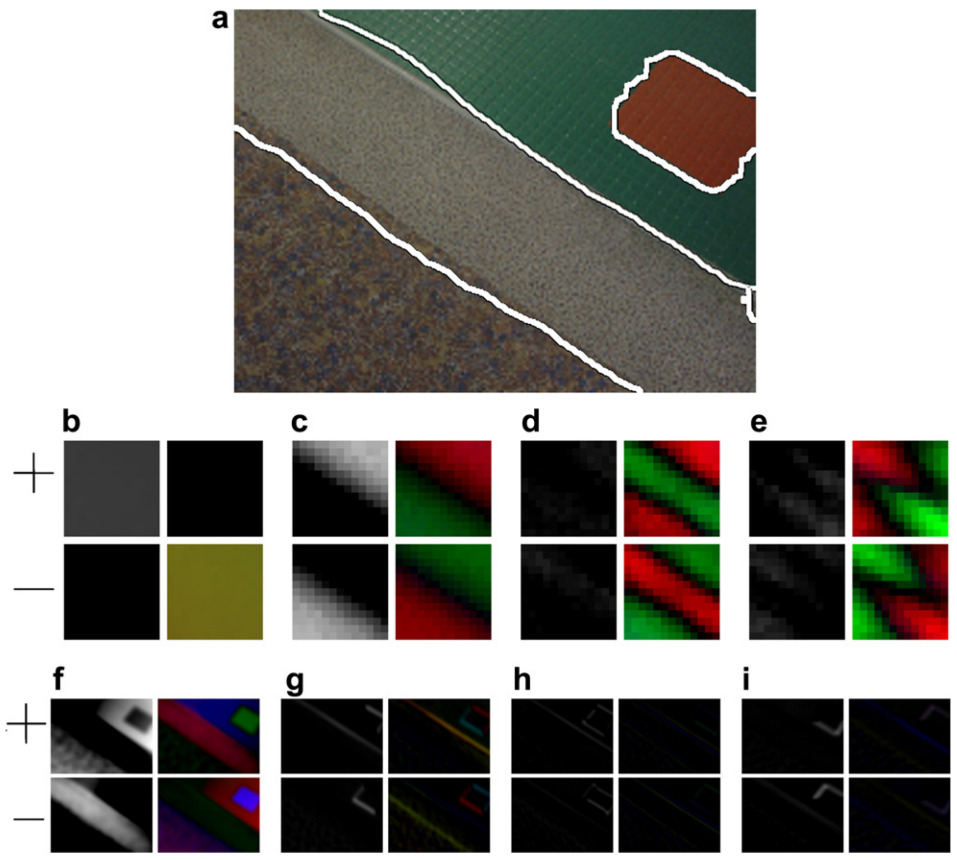

Fig. 8. The quaternion texture basis of one image: (a) shows the segmentation; (b–c) show the positive (upper row) and negative (lower row) components of the real and imaginary parts of the first 4 QPCA basis vectors obtained from $13 \times 13$ subwindows. The real part of a quaternion basis vector is represented as gray scale image, and the imaginary parts as RGB color image. Using the same encoding, (f–i) show the first four texture feature maps. The majority of the energy is in the first component.

larger than needed and the extra vectors discarded at the very end.

## 5. Experiments

The quaternion texture segmentation method was tested on input images of approximately 36,864 (192 by 192) pixels. Larger images are resized to these approximate dimensions while preserving their aspect ratio. The subwindow size was 13 by 13 with a 6 pixel overlap horizontally and vertically. As a result, the total number of subwindows for an image of size 192 by 192 equals 676. Since the total number of feature vectors is 676, the size of each subwindow texture sample is 169, and each pixel is represented by a 4-dimensional quaternion number, the total dimension of the input is $169 \times 36864 \times 4$. Memory requirements make this impractical for the QSVD algorithm and completely unmanageable if the input image size or the number of subwindows is increased. When the memory requirements become too large, we use IQPCA instead of QPCA but prefer QPCA when feasible.

After the quaternion eigenbasis is calculated, the question is then how many basis vectors to use as texture feature descriptors. Preliminary experiments showed that in practice it was best to use just the first QPCA basis vector. For the subsequent clustering and segmentation stages, the first basis vector appears to represent just the right amount of detail about the textures. As a result, a subwindow's texture is represented by a single quaternion obtained as the dot product of the subwindow with the first basis vector.

Examples of the new quaternion texture segmentation method are illustrated in Figs. 3–8. In Fig. 3, the original input image from Hoang et al. [8] contains five color textures of varied color and pattern. The left hand quadrants have similar color and texture, although the texture is at a different orientation, while the upper quadrants have exactly the same pattern but different color. The central circular region has a different color and texture. Our new method successfully segments the five regions as shown in Fig. 3b. The results can be compared to Hoang's [8]. Not shown in Fig. 3 is what happens as we lower the similarity threshold so more regions merge: First, the left hand quadrants, which are similar in color and texture, merge; Second, the shadow area in the lower-right quadrant disappears. With the quaternion representation, it is also possible to change the relative importance assigned to color versus grayscale texture. When the segmentation is based on intensity alone, the upper quadrants always merge. On the other hand, segmentation based on color alone always merges the left quadrants.

Fig. 4 compares the quaternion segmentation to the Markov random field segmentation [11]. The comparison is hampered by the fact that the input image is based on a scan of the published version of the original input image rather than the original data file, which is no longer available. Further results on real images are shown in Fig. 5. Figs. 6 and 7 show how the results vary when a smaller subwindow size is used and when the threshold for region merging is varied. Fig. 8 displays the first four QPCA basis vectors and the four corresponding feature images obtained by projecting the original textures onto each of these basis vectors. The real component of a quaternion vector is represented as a gray-scaled image, while the *i*, *j*, and *k* components of a quaternion vector are represented as an RGB color image. The '+' and '−' signs labeling the rows indicate that the image shows only the positive or negative pixels. Pixels not corresponding to the sign are zeroed and appear as black. As can be seen in Fig. 8(b), the first basis vector is much smoother than the others. The basis vectors in Fig. 8(d) and 8(e) are more representative of color edges. As a consequence, the image representing the quaternion texture map (Fig. 8f) contains significant color and texture information because the first basis vectors is the one accounting for the greatest variance.

The total computation time required to segment a 192-by-192 image using a 13-by-13 subwindow for every pixel is 100 s using Matlab on a P4 1.6G CPU with 2G memory. It requires 5 s to compute the quaternion eigen-bases of the training set using IQSVD. The remaining time is spent on projecting each texture subwindow (one per pixel) onto the quaternion basis vector (13 s), classifying (18 s), and merging (64 s). The computation time for quaternion texture segmentation might be reduced by using smaller training sets, since when the training set is small, it takes less time to compute the eigen-basis and to construct the classifier; however, there may be a trade-off in terms of accuracy. The filtering step can be sped up when the texture subwindow size is large by performing the convolution in Fourier space using the quaternion Fourier transform [5].

## 6. Conclusion

Quaternions were used here to represent color in the context of the automatic analysis of color texture. Color texture features are determined by quaternion principal components analysis of the data from many sub-windows of an image. The use of quaternions means that overall the proposed method is much less complicated than for example that proposed in [8]. Given a quaternion operation tool package, our method is straightforward to implement. In addition, the number of user-controlled parameters in our algorithm is fewer. The overall accuracy of the methods, however, is comparable. The quaternion representation of RGB color texture nicely encodes spatial intra-channel and inter-channel relationships.

An incremental QPCA algorithm is introduced so that larger training sets can be processed. Quaternion texture analysis provides a simple and effective way of combining the color and spatial texture features. The texture feature maps show that the most significant features are contained in the first quaternion eigenbasis. Experiments show that these features can be used to cluster textures and then segment images into regions of homogenous texture. This demonstrates once again that the quaternion representation of color, which treats color channels as single unit instead of as separate components, truly is effective.

## Acknowledgment

This work supported by the Natural Sciences and Engineering Research Council of Canada.

## References

[1] S.E. Grigorescu, N. Petkov, P. Kruizinga, Comparison of texture features based on Gabor filters, IEEE Trans. Image Process. (11), No. 10, October 2002, 1160–1167.
[2] S. Arivazhagan, L. Ganesan, Texture segmentation using wavelet transform, Pattern Recogn. Lett. (24), No. 16, December 2003, 3197–3203.
[3] Z. Lu, W. Xie, J. Pei, J. Huang, Dynamic texture recognition by spatiotemporal multiresolution histogram, Proc. IEEE Workshop on Motion and Video Computing (WACV/MOTION'05) (2005) 241–246.
[4] C.J. Evans, T.A. Ell, S.J. Sangwine, Hypercomplex color-sensitive smoothing filters, IEEE Int. Conf. Image Process. (ICIP), 2000, I, 541–544.
[5] S.J. Sangwine, T.A. Ell, Hypercomplex Fourier transforms of Color images, in: IEEE Internat. Conf. on Image Processing (ICIP 2001), Thessaloniki, Greece, 2001, I, pp. 137–140.
[6] S.C. Pei, J.H. Chang, J.J. Ding, Quaternion matrix singular value decomposition and its applications for color image processing, in: IEEE Internat. Conf. on Image Processing (ICIP) 2003, I, pp. 805–808.
[7] N. Le Bihan, S.J. Sangwine, Quaternion principal component analysis of color images, IEEE Int. Conf. Image Process. (ICIP) (2003) 809–812.
[8] M.A. Hoang, J.M. Geusebroek, A.W.M. Smeulders, Color texture measurement and segmentation, Signal Process. 85 (2005) 265–275.
[9] H.T. Nguyen, M. Worring, A. Dev, Detection of moving objects in video using a robust motion similarity measure, IEEE Trans. Image Proc. 9 (2000) 137–141.
[10] D. Alleysson, S. Süsstrunk, Spatio-chromatic PCA of a mosaiced color image, in: Proc. IS&T Second European Conference on Color in Graphics, Image, and Vision (CGIV 2004), April 2004, pp. 311–314.
[11] D. Panjwani, G. Healey, Markov random field models for unsupervised segmentation of textured color images, IEEE Trans. Pattern Anal. Mach. Intell. 17 (1995) 939–954.
[12] P.M. Hall, A.D. Marshall, R.R. Martin, Incremental eigenanalysis for classification, in: Proc. British Machine Vision Conference, PH Lewis and MS Nixon, Eds., 1998, pp. 286–295.
[13] B.S. Manjunath, S.Chandrasekaran, Y.F. Wang, An eigenspace update algorithm for image analysis, in: Proceedings of International Symposium on Computer Vision—ISCV, Coral Gables, FL, USA, November 1995, pp. 551–556.