Illumination–Invariant Image Retrieval and Video Segmentation

Mark S. Drew, Jie Wei, and Ze-Nian Li
School of Computing Science, Simon Fraser University,
Vancouver, B.C. Canada V5A 1S6
email {mark, jiew, li}@cs.sfu.ca

Abstract

Images or videos may be imaged under different illuminants than models in an image or video proxy database. Changing illumination color in particular may confound recognition algorithms based on color histograms or video segmentation routines based on these. Here we show that a very simple method of discounting illumination changes is adequate for both image retrieval and video segmentation tasks. We develop a feature vector of only 36 values that can also be em used for both these objectives as well as for retrieval of video proxy images from a database. The new image metric is based on a color–channel–normalization step, followed by reduction of dimensionality by going to a chromaticity space. Treating chromaticity histograms as *images*, we perform an effective low–pass filtering of the histogram by first reducing its resolution via a wavelet–based compression and then by a DCT transformation followed by zonal coding. We show that the color constancy step — color band normalization — can be carried out in the compressed domain for images that are stored in compressed form, and that only a small amount of image information need be decompressed in order to calculate the new metric. The new method performs better than previous methods tested for image or texture recognition and operates entirely in the compressed domain, on feature vectors. As well as achieving illumination invariance for video segmentation, so that e.g. an actor stepping out of a shadow does not trigger the declaration of a false cut, the metric reduces all videos to a uniform scale. Thus thresholds can be developed for a training set of videos and applied to any new video, including streaming video, for segmentation as a one–pass operation.

1 Introduction

Color histograms play a pivotal role both for image retrieval from an image database [1] and for segmentation of videos (see, e.g., [2]). As well, we may wish to use such feature vectors for characterization of video proxies such as keyframes in order to efficiently browse a video database.

However a problem arises if illumination differs for model objects and test images. RGB values can be strikingly different when images arise from different sources that include changes in illumination or a different camera. Several schemes have evolved for discounting changes in illumination – the *color constancy* problem. Here we do not mean to provide a comprehensive solution to this difficult problem but instead wish to utilize a method for color constancy that is very simple while effectively discounting illumination to a degree adequate for the tasks addressed.

The simplest previous illumination–invariant method is based on the angles between color image channels [3]. In that method color constancy is approximated by first normalizing (mean–subtracted) color channels to unit length. Here we adopt this first stage (for non–mean–subtracted color channels) as an approximate method for discounting illuminant change. Below we show that in fact this same approach can, to some extent, remove dependence on just what camera was used in producing images, as well.

Here, we combine a technique of first applying a wavelet–based reduction with a second step of truncation via zonal coding of the Discrete Cosine Transform (DCT) image. This results in an effective low–pass filtering of the chromaticity histogram. The resulting image indexing scheme is very efficient in that it uses a feature vector of only 36 or 72 values. The method is remarkably accurate for such a simple, compact scheme and, as we show below, is more resilient to noise than other methods explored here.

The guiding principle for our algorithm is simplicity and speed. Nevertheless we found that bringing appropriate complexity to bear, when it is needed, produced better performance; we used wavelet–based compression to reduce the histogram size to 16×16 because that produced a more effective descriptor than simply using coarse uniform bins (cf. e.g. the $8 \times 8 \times 4$ bins in [4]). This way, we could use full resolution in chromaticity space and not have to deal with color clustering [5]. At the same time, however, the method has simplicity in its color constancy scheme and in the straightforward L1 or L2 metric used for matching.

In §2 we outline the significance of image normalization and the role of chromaticity histograming in the illumination–invariant characterization of color images. We describe an implementation using a histogram–image compression scheme employing wavelet–based reduction, DCT, and zonal coding. Experimental results for image indexing using the new image difference metric are presented in §3.

In §4 we show how the illumination–invariant image difference metric can be applied to the task of video temporal segmentation. This is a fundamental task in video understanding; it can be compared to the step of finding tokens as a first stage in lexical analysis. Video segmentation is crucial for the ensuing video indexing of a digital library or storyboarding of digital videos. The new metric allows for the judicious creation of a set of thresholds, for frame differences of 32, 16, 8, 4, 2, and 1 frames, using robust statistics offline. This allows for a binary search for cuts, for any new video, without having to individually characterize each new video. Video segmentation results are presented in §5 and the paper concludes with some general observations in §6.

2 Theory: Illumination–Invariant Image Characterization

2.1 Color–Channel–Normalized Images and Chromaticity Histograms

Below we set out the rationale for the color–channel–normalization color constancy strategy we adopt, and then show how images can be characterized using compressed chromaticity histograms.

2.1.1 Illumination invariance and image normalization

Finlayson et al. proposed indexing on six numbers corresponding to the "angles" amongst image and edge–image channels [3]. That method consists of the following: *normalize* each mean–subtracted color channel R, G, B to length 1 by dividing by the square root of the sum of squares of that channel, and then take as indexing numbers the three angles formed from the inverse cosine of the products $R \cdot G$, $R \cdot B$, $G \cdot B$. Along with these, append a second set of angles derived in exactly the same way from the edge image of the smoothed color–normalized

color image, $\nabla^2 G\boldsymbol{\rho}$, where $\boldsymbol{\rho} = (R, G, B)$. The idea is that, if camera sensors are sufficiently narrowband, these angles are invariant to color shifts in the illuminant because in that situation illumination change corresponds to a simple scaling of the color channels (the "coefficient rule"). In the limit of Dirac delta function sensors, this approximation would hold identically. If sensors are not "sharp" enough, then provided one knows the camera sensor curves one can carry out a "spectral sharpening" operation [6] to bring the color angle approximation more in line with actual conditions [3]. Thus illumination change amounts to a diagonal matrix transform among color channels. Even without sharp sensors, if the illuminant is fairly white then a factor model [7] of color formation still leads to a diagonal transform.

To see how a diagonal model of illuminant change arises, consider the RGB triple $\boldsymbol{\rho}$ resulting from some set of lights with spectral power distributions $E_i(\lambda)$ and indexed by $i = 1..L$ impinging on a Lambertian surface with surface spectral reflectance function $S^x(\lambda)$, under conditions of orthography. If the camera system color sensors have sensitivity functions $\boldsymbol{q}\,(\lambda)$ then

$$\boldsymbol{\rho}^{\,x} \;=\; \sum_{i=1}^{L} (\boldsymbol{a}_{\,i}^{\,T}\, \boldsymbol{n}^{\,x}) \int E_i(\lambda)\, S^x(\lambda)\, \boldsymbol{q}\,(\lambda)\, d\lambda \tag{1}$$

where $\boldsymbol{n}^{\,x}$ is the surface normal for that pixel and light E_i has normalized spatial direction $\boldsymbol{a}_{\,i}$. If camera sensors $\boldsymbol{q}$ are narrowband enough to be approximated as sampling a single frequency,

$$q_k(\lambda) \simeq q_k(\lambda_k)\delta(\lambda - \lambda_k) \quad , \quad k = 1..3$$

with constants $q_k(\lambda_k)$, then we have

$$\rho_k^x \;=\; \tau_k^x S^x(\lambda_k) q_k(\lambda_k) \tag{2}$$

where

$$\tau_k^x \;=\; \sum_{i=1}^{L} (\boldsymbol{a}_{\,i}^{\,T}\, \boldsymbol{n}^{\,x})\, E_i(\lambda_k)$$

Clearly, under a change of illumination environment $E_i \rightarrow E_i'$, $\boldsymbol{a}_{\,i} \rightarrow \boldsymbol{a}_{\,i}'$, $L \rightarrow L'$, we have a diagonal transformation of RGB values $\boldsymbol{\rho}$ *provided the surface is flat* so that τ_k is independent of x. Then the normal $\boldsymbol{n}$ of the flat surface patch can be tilted, in the new image under new lighting, and still have a diagonal transform relate color under the old illumination to color under the new illumination with the diagonal elements τ_k'/τ_k being the same for every pixel. Ignoring surface normal terms or considering very distant viewing as in satellite imaging (cf. [8]) constitutes a good working hypothesis, at least for image retrieval purposes, if not for physically correct surface reconstruction, and we see below in §4 that indeed applying the diagonal model of illuminant change works well enough for this purpose, even for non–flat objects. Even changing the camera itself stills results in another diagonal transform under this assumption, provided that both cameras have sensitivity peaks roughly in the same area of the sprectrum and are sufficiently narrowband. In practice these requirements are approximately met even for real camera sensitivities.

Another argument that has been used for a diagonal model is that surface spectral reflection can be approximated as a dimension–3 expansion in a finite–dimensional–model (FDM) basis set [9]. However even without an FDM or sharp sensors, a factor model of color [7] also gives a decomposition similar to eq. (2). In computer graphics a simple approach to approximating color for reflected light consists of multiplying the RGB triple for the illuminant times that for the surface. In terms of eq. (1) this amounts to approximating the color b_k^i of light $E_i(\lambda)$ reflected from surface $S(\lambda)$ by

$$b_k^i \;\simeq\; s_k e_k^i/\sigma_k \;, \quad k = 1..3 \tag{5}$$

where s_k is the surface color under equi-energy white light

$$s_k \;=\; \int S(\lambda) q_k(\lambda) d\lambda \tag{6}$$

and e_k^i is the color of the ith illuminant,

$$e_k^i \;=\; \int E^i(\lambda) q_k(\lambda) d\lambda \tag{7}$$

The camera scaling term is

$$\sigma_k \;=\; \int q_k(\lambda) d\lambda \tag{8}$$

Borges [7] carefully considered this approximation and showed that it is accurate provided the illuminant is "white enough".

Substituting into eq. (1) we thus have

$$\boldsymbol{\rho}^{\,x} \;=\; \tau_k^x \, s_k^x \,/\, \sigma_k \tag{9}$$

where now τ_k^x is given by

$$\tau_k^x \;=\; \sum_{i=1}^{L} (\boldsymbol{a}_i^{\,T} \boldsymbol{n}^{\,x}) \, e_k^i \tag{10}$$

and again a diagonal transform under a lighting change holds approximately.

Normalization of each color band in an L2 norm (i.e., Euclidean distance) effectively enforces illumination invariance, in a diagonal model: for considering each color channel $\rho_k, k = 1..3$ separately as a long vector (cf. [3]) we see that an illumination change

$$\rho_k^x \;\rightarrow\; \rho_k^{x\prime} \;=\; (\tau_k'/\tau_k)\, \rho_k^x \tag{11}$$

simply corresponds to a change of vector length. Normalizing these lengths to unity for every image effectively discounts change of lighting. In fact, color band normalization can be in an L2 norm or an L1 norm, i.e., dividing by the sum of each channel. As well, one can divide by the mean of each channel, thus removing the effect of image size.

In [10] we compare using normalized color bands with other ostensibly color constant indexing methods and show that color constant image indexing can be accomplished accurately using compressed histograms of the *chromaticity* (r, g) for each pixel, defined as [11]

$$r \;=\; R/(R+G+B) \quad , \quad g \;=\; G/(R+G+B) \tag{12}$$

provided R, G, and B are separately scaled to length 1. We define the 2–channel chromaticity image as that $N \times 2$ array formed from the color channels of an N–pixel image. The chromaticity is the projection of an RGB triple onto the planar triangle joining unit distance along each of the color axes if intensity is removed by dividing by the sum of the three color bands. Here, chromaticities are formed using the *normalized* color image bands R, G, B, to compensate for the possibility of changing illumination between model and test image or as a video progresses and also to be able to index into a database of videos that are similar and yet were shot under different lighting conditions. An advantage of using the chromaticity is that since the range of possible values is necessarily in the interval $[0, 1]$ for every input image, we are not faced with picking a new threshold for determining video cuts when an image sequence happens to have a different maximum value than another one or has images of different resolution. Instead, we may form histograms of the chromaticity images having unit volume and always fall into the same range. The color–channel–normalization step ensures that an object imaged under different illuminants is approximately independent of the illuminant. [1]

Fig. 1(a) shows a color image and Fig. 1(c) shows the same object imaged under a different lighting environment. Fig. 1(b,d) show the chromaticities for these two images, displayed by plotting r versus g. In the end, we wish to make histograms that capture the information in plots like these.

Fig. 1(e) shows the same type of scatterplot as Fig. 1(b), but for first image, Fig. 1(a) once color-channel-normaliztion has been applied; and Fig. 1(f) does the same for the second image. Without the normalization there is a shift in the plots of chromaticities for the two images (in actuality a fairly complex change, not just a translation). We can see that for normalized images the plots do not shift, and therefore color–channel–normalization in effect provides a simple color constancy preprocessing step.

[1] Note that color band normalization can be in an L2 norm or an L1 norm, i.e., dividing by the sum of each channel. As well, one can divide by the mean of each channel, thus removing the effect of image size.

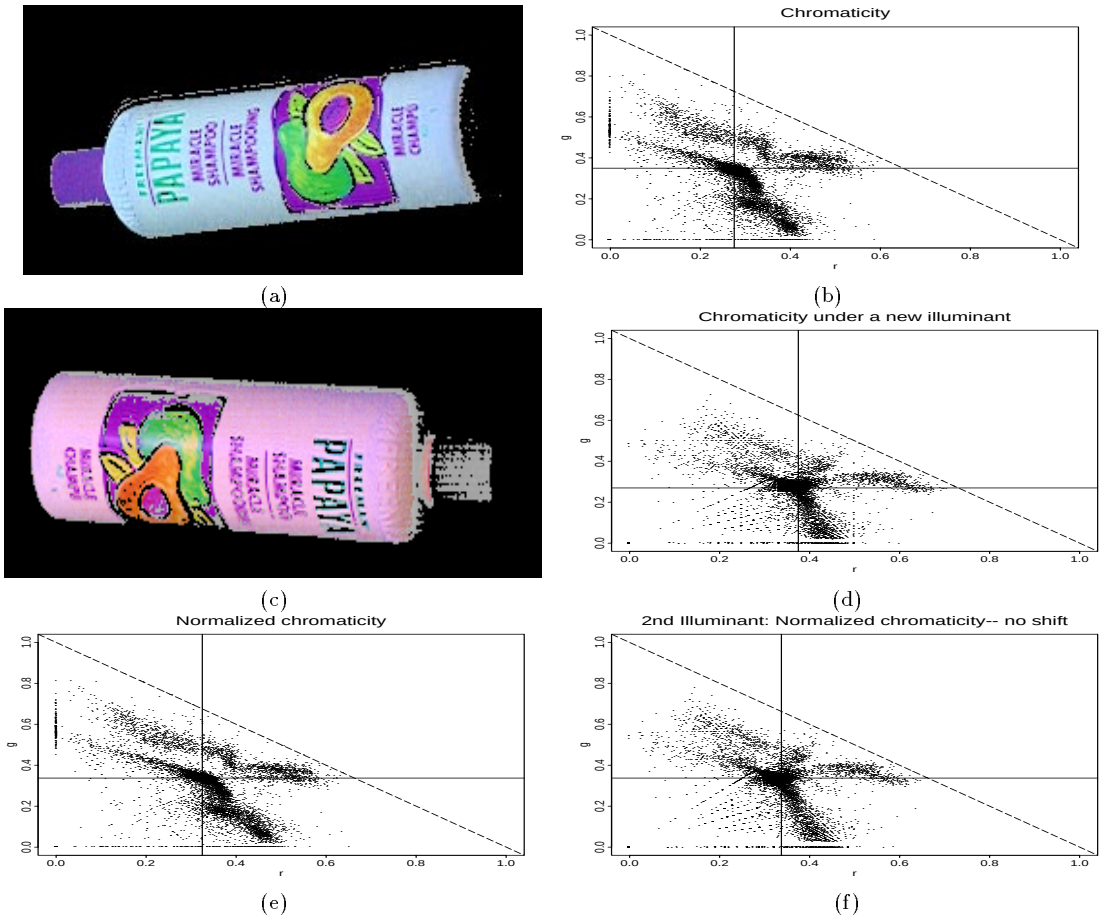


Figure 1: (a): Model image. (b): Chromaticity for unnormalized model image. (c): Test image, under new illumination. (d): Chromaticity for unnormalized test image. (e): Normalized chromaticity for first illumination. (f): Normalized chromaticity for second illumination – no shift.

2.1.2 Histogram intersection

In Swain and Ballard's seminal work on color histograms [1] 3–dimensional histograms are used in which each of the three color bands is divided into sixteen bins over the range 0..255. Further, Swain and Ballard use a matrix–transformed color space as well as RGB and more coarsely sample the first dimension, for a total of $8 \times 16 \times 16 = 2048$ bins.

Here we can define 2–dimensional histograms indexed by float values of the r and g chromaticity values. Swain and Ballard's very useful histogram intersection statistic, equivalent to unity minus an L1 metric, becomes the following for the intersection of chromaticity histograms H_a and H_b:

$$\mu \equiv \sum_{i,j} min\{H_a(i,j), H_b(i,j)\} \tag{13}$$

Swain and Ballard normalize intersection (or match) values by the number of pixels in the model histogram, and thus matches are between 0 and 1. Alternatively, one can make the volume under each histogram equal to unity, effectively making each image have the same number of pixels and turning the histogram into a probability density. We computed matches using the above method, and also using an L2 distance and found little change in match rankings amongst images in the database when the algorithm was presented with a novel image. For compressed histograms (below), we adopt an L2 distance measure since negative numbers are involved.

We can use any convenient size of chromaticity histogram, resolving chromaticity floating point numbers as finely as needed, and we found that histogram size has an appreciable but not dramatic effect on the effectiveness of the image similarity metric. *Treating each chromaticity histogram as an image*, in §4 we reduce its size by a wavelet–based compression and go on to perform a DCT and zonal truncation that results in a feature vector of only 36 values.

As well, since only half the histogram bins are needed, i.e., those corresponding to the lower–left half since by the definition of the chromaticity we have $r + g \leq 1$, we can start with a 32×32 histogram and fold it into a 32×16 one, as shown in [10]. Then interpreting this as a pair of 16×16 histograms we can develop an image difference metric based on a length–72 feature vector. Both schemes produce comparable results, with the 72–component scheme being slightly better.

2.1.3 Other illumination–invariant methods

The color angle idea can be summarized as a use of a covariance matrix (plus the addition of an edge image covariance). As such, its closest relative is the method of Healey and Wang [12], which has been applied to color textures. Histogram–based methods have distinct advantages over [12], however, in that they are invariant to 2D rotations and translations.

Healey and Wang's method [12] consists of considering a set of correlations $R_{jk}(n,m)$, $j,k = 1..3$ (i.e., covariance terms for mean–subtracted image channels), with shifts n, m from 0 to 16 pixels. In practice, they use $n = 0..16$ and $m = -16..16$, so for each set of channel products jk they use 561 correlations. Then there are 6 different color channel products (RR, RG, etc.) for each set of shifts, so an image feature vector has $561 \times 6 = 3366$ elements. For matching, an error measure is taken to be the L2 distance between a new image's feature vector and the result of writing that vector in terms of the eigenvectors of each model image's set of correlations. Let us refer to this method as the Correlation Method.

In comparison, other illumination–invariant matching schemes include the color–constant–color–indexing (CCCI) method [13], which uses the counts in 4096 histogram bins for a color edge image, and the method of Slater and Healey [14], which uses moments of color histograms and makes the assumption that local image patches can be modeled as textured, flat planes.

Another approach that has been employed is to use *band ratios* R/B, G/B [15]. Pioneering work on using ratio images was carried out by Skifstad and Jain [16] in the context of change detection between temporally separated grayscale images. For the case of color images, suppose only a single illuminant is present; then any ratio of color bands removes the shading term $\boldsymbol{a} \cdot \boldsymbol{n}^{\,x}$ [17]. Lee et al. [15] employ this property along with the diagonal model: assuming illuminant change induces a diagonal multiplication $\rho_k \rightarrow \rho'_k = C_k \rho_k, k = 1..3$ they recover constants $log(C_k/C_j)$ via a Hough–like voting procedure using the Fourier transform of histograms of band–ratio images ρ_k/ρ_j. However we found that the chromaticity (12) gave better performance (our approach was not to attempt

to recover the C_k but instead to simply set $logC_k$ to zero by subtracting the mean of log band–ratio images). The increased performance of chromaticity over band ratios may be due to the increased robustness of the sum in the denominator of eq. (12) in the presence of noise.

2.2 Chromaticity Histogram Compression

Here we show that if we store a representation of each histogram that is first reduced in size by a wavelet transformation, and then further reduced by going to the frequency domain and discarding higher–frequency DCT coefficients, we can derive a simple indexing method that is efficient and invariant under illuminant change. In the following method, the number of parameters in the feature vector is only 36 or 72 values.

Moreover, we show that the essential color constancy step, that of normalizing image color bands, can be carried out either on uncompressed images or directly on the DCT transform of a color image. Since an orthogonal transform conserves energy, we may derive an L2 norm from the DCT coefficients of a block–transformed image stored in the JPEG format. Alternatively, since the DC component stores block–average information, we can instead perform an L1 normalization directly in the DCT domain.

2.2.1 Compression of histograms treated as images

Chromaticity histogram matching without compression could be computationally intensive; we would like to recover an accurate approximation of histograms without sacrificing efficiency. As a guiding principle it would also be sensible to maintain a linear relationship between the histogram and its compressed representation. We can adhere to this principle by applying only linear operations while compressing the histograms. Therefore, here we first apply a linear low–pass filter to both histograms, resulting in new histograms $\boldsymbol{H}$ and $\boldsymbol{H}'$. To best approximate the chromaticity histograms the low–pass filtered histograms should approximate the original ones as closely as possible, yet be of lower resolution. The scaling function of biorthonormal wavelets, as a symmetrical low–pass filter, can be exploited to that end. Basically, scaling functions with more "taps" use polynomials of higher order to approximate the original function (the number of taps is the number of nonzero coefficients) [18]. Our main concern is to capture the most detail but in lower resolution, and after some experiments we arrived at a good balance between efficiency and precision by using the symmetrical 9–tap filter [19]. The 1D mask of the separable 2D scaling function is:

$$\{0.037829, -0.023849, -0.110624, 0.377403, 0.852699, \\ 0.377403, -0.110624, -0.023849, 0.037829\}$$

After applying the scaling function several times to the original histograms, assuming for simplicity square histograms with resolution $2^n \times 2^n$, we obtain size 16×16 lower resolution histogram images.

We then apply a DCT, transforming $\boldsymbol{H}$ into $\widehat{\boldsymbol{H}}$ and indexing on $\widehat{\boldsymbol{H}}$. Since the lower frequencies capture most of the energy of an image, after applying the DCT we can retain just the lower frequency coefficients for histogram database indexing — a very effective and efficient way of realizing a further low–pass filtering. By experiment we found that using only 36 coefficients worked well, these being those in the first 36 numbers in the upper left corner of the DCT coefficient matrix, as shown in Fig. 2. [2]

The reason for not simply continuing the wavelet compression to an even smaller size is that the DCT coding allows us to concentrate on the part of the histogram where the information actually is, as well as allowing us to tune just how many zigzag diagonals to retain.

Denote by $\boldsymbol{H}_d$ the 36 values (derived from DCT coefficients) in the order illustrated in Fig. 2. We index on the L2 distance between $\boldsymbol{H}_d$ for the model histogram and that for the test histogram. Let us call the chromaticity–histogram method (with no compression) the *ChromHist–A method*, and that based on compressed histograms the *ChromHist–B1 method*.

Populating the database, then, consists of calculating off-line the 36 values $\boldsymbol{H}_d$, viewed as indices for each model image. For image query, first the 36 values for the query image are computed, thus obtaining $\boldsymbol{H}'_d$; then for

[2] Instead of using a conventional 8×8 window for the DCT, a 16×16 window is adopted. As a result, a finer resolution (twice as high as with 8×8) in the spatial frequency domain is realized. Since the low–pass filtering after DCT can only retain a limited number of coefficients for efficiency, the net effect of having a larger (16×16) window is that a more detailed parameterized description at the lower end of the spectrum is facilitated. This is beneficial when very low–resolution wavelet images are used for matching in our method.

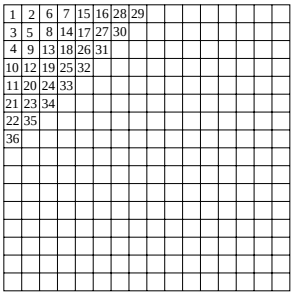


Figure 2: Arrangement of DCT coefficients

every model image, the L2 distance $[\sum(\boldsymbol{H}'_d - \boldsymbol{H}_d)^2]^{1/2}$ is calculated. The model image minimizing the distance is taken to be a match for the query image.

Note that in the ChromHist–B1 method only reduced, DCT transformed, zonal coded histograms are used — no inverse transforms are necessary and the indexing process is entirely carried out in the compressed domain.

The choice of a reduced resolution of 16×16 for chromaticity histograms and 36 DCT coefficients retained was made quite empirically. For comparison we also define a *ChromHist–B2 method* in which the wavelet–reduced chromaticity histogram images have a resolution of 32×32. For this method we could still use a 16×16 window for the DCT by applying the DCT to each of the four quadrants of the 32×32 histogram. However, a more compact representation can be used here. Recall that only the lower left–half of the chromaticity histogram contains non–zero counts (shown as the shaded region in Fig. 3(a)). To reduce computation we rearrange the histogram by cutting off the right–half of the shaded triangle and append it to the top of the remaining region to get a new rectangular region shown in Fig. 3(b). The new histogram consists of two 16×16 subimages; applying a 16×16 DCT to each of these, with 36 coefficients each, gives method ChromHist–B2, which indexes on a total of 72 values.

As well, an auxiliary step of carrying out a square root operation on the histogram values before turning them into probability densities was found to improve the performance over the results presented in [20] where no square root was applied.

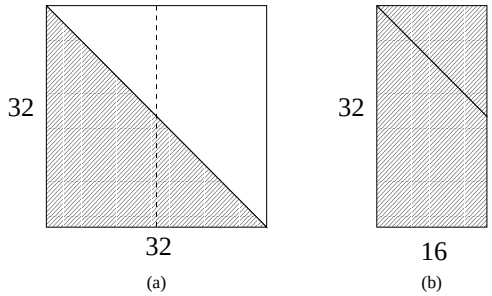


Figure 3: ChromHist-B2: Preparing two 16×16 subimages

2.3 Normalization in the DCT domain

2.3.1 Using partial images stored in compressed form

If images are stored in the JPEG format or for videos stored as motion–JPEG or MPEG [21] then it is straightforward to undo the final stages of run–length encoding and Huffman coding so that they are in the form of (quantized) DCT coefficients. Accounting for chroma subsampling and the matrix transform to YCbCr, it is straightforward to work with this version of DCT images of color bands. We can carry out an L2 normalization directly in this space, since by Parseval's theorem the sum of squares of the pixel values equals the sum of squares of the DCT coefficients. Since we are interested in algorithm economy, we can further gain by reconstituting the normalized color image using only a few DCT coefficients before forming a chromaticity histogram and carrying on with the algorithm. Accordingly we denote as *method ChromHist–C* the notion of normalizing in the DCT domain and performing an inverse transform on only a few terms.

As another alternative, we can produce an image norm using only those low–frequency components we mean to retain in the color image. Let us denote as *method ChromHist–C1* the first approach, wherein normalization of

the full image is carried out in the DCT domain and then only a few coefficients are reconstituted. And denote as *method ChromHist–C2* the second approach, wherein only a few coefficients are used to contribute from each block to the image norm, and then those components are used in the DCT transform.

Both approaches amount to an image blurring (an almost free one, assuming images to be already stored in DCT form). As it turns out, in the databases examined below, the second method, ChromHist–C2, gives better results.

Note that while we are motivating the idea of carrying out a normalization in the DCT domain by our color constancy argument of §2.1.1, if illumination change amounts to a diagonal transform then this is true in the DCT domain as well— illumination change corresponds to multiplying the DCT coefficients for the three color bands by a triple of numbers. Therefore as long as we divide DCT coefficients by any value that includes those multipliers we arrive at illumination–invariant descriptors for images.

2.3.2 L2 and L1 normalizations in the DCT domain

Let us examine image indexing when the images themselves are already stored in a form that readily converts to blockwise DCT images. It makes sense to normalize color bands in the DCT domain (of the color images, not histograms made from these); then any further calculations can be carried out in the DCT domain as far as possible.

We can use either an L2 norm, using all the DCT values, or else normalize in an L1 norm, using just the DC coefficients in each block. Another, more efficient alternative for the L2 norm would be to use only a few DCT coefficients, approximating the image norm using only a smaller number of terms.

Since DCT coefficients for the *image* are available we can not only normalize in the DCT domain but we can also choose to reconstitute color images by inverting the DCT transform for just a few of the zigzag diagonals in Fig. 2 (method ChromHist–C1). This method says: we mean to use normalized bands, so we normalize using all N^2 DCT coefficients in each block (with $N = 8$, e.g.), summing over blocks to produce a color band squared length. This is not really any saving over normalizing in the spatial domain. However, once we have normalized let us reconstitute the image using only lower–frequency terms, before going on to create histograms.

Method ChromHist–C2 results from forming the image norm using only a few coefficients. This method says that we'll content ourselves with a blurred version of the image, with higher frequencies removed, and then apply the color constancy strategy. This method does indeed produce a saving in forming the norm, since only a few DCT coefficients contribute to it; and as well only those coefficients are used in reconstituting the color image.

As it happens, for the L1 norm both method ChromHist–C1 and ChromHist–C2 are identical. The reason is that the L1 norm for an image block is the same whether or not we use all N^2 DCT coefficients or we include only $d < N^2$ coefficients. To see this, suppose we use zigzag–ordered coefficients $z = 1..d$. Then with frequency variables $u = u(z), v = v(z)$ for block size N the L1 block norm that we require is that for the image that would be reconstituted: we need the space–domain sum $\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}$ of the DCT–inverted DCT coefficients that are retained:

$$L_1 = (2/N)\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left\{\sum_{z=0}^{d-1}\Lambda\left(u(z)\right)\Lambda\left(v(z)\right)\cos\left(\frac{\pi u}{2N}(2i+1)\right)\cos\left(\frac{\pi v}{2N}(2j+1)\right)F\left(u(z),v(z)\right)\right\}$$

$$= (2/N)\sum_{z=0}^{d-1}\Lambda\left(u(z)\right)\Lambda\left(v(z)\right)F\left(u(z),v(z)\right)\left\{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\cos\left(\frac{\pi u}{2N}(2i+1)\right)\cos\left(\frac{\pi v}{2N}(2j+1)\right)\right\} \qquad (14)$$

where $\Lambda(u) = 1/\sqrt{2}$ if $u = 0$ and 1 otherwise. But the space-domain sum over i, j, above in braces, equals N^2 if $u = v = 0$ and 0 otherwise, so the L1 norm is still just $NF(0,0)$, exactly the same as if we had used all $z = 0..(N^2 - 1)$ DCT coefficients. I.e., this simply means that if we choose to DCT–invert an image for which we have zeroed out some coefficients, then we still get an image with the mean as specified in the DC term. The result then is that whether we divide by the "full" L1 norm in the DCT domain [dividing by $F(0,0)$] and then truncate the zigzag coefficients and DCT–invert, which is method –C1, or if we truncate, normalize, and invert, method –C2, we end up with exactly the same normalized image.

The situation is different for the L2 norm, however, and as a sub–result to Parseval's theorem we arrive at an L2 norm

$$(L_2)^2 \;=\; \sum_{z=0}^{d-1} F^2\left(u(z), v(z)\right) \tag{15}$$

I.e., normalizing in the DCT domain using all the DCT coefficients and then inverting using only a few DCT values, method –C1, and on the other hand calculating a norm using only the first d coefficients as above and then again inverting using the d values, method –C2, give different normalized images.

3 Image Indexing Results

3.1 Swain's database: no illumination change

3.1.1 Full images

Here we compare the new method (denoted ChromHist–A), based on the chromaticity of color–normalized images, and its variants that operate in the compressed domain (denoted ChromHist–B1 and B2) with the original Color Indexing method [1], and with the Color Angles method [3], with model and test images taken from Swain's original paper. We find that the new method does well on this database, better than the other methods tested. Since the original images are quite special in the sense that objects appear on a black background that is easily thresholded away, we also test how the performance of each method degrades in the presence of clutter. A simple approach to providing a background for each image is to generate random noise all over the image—this also provides a standard partial sensitivity analysis. To the eye, this makes no difference in recognizing correct matches.

The new ChromHist method is relatively insensitive to the size of the histogram. Here we use 128×128 histograms; this is not necessarily optimum, but is useful for wavelet–based compression because each side of the histogram 'image' is a power of 2.

Table 1 shows matching scores for the methods tested, on the original images and on images with uniform random noise with rms value 15 added. A rank of 1 means a correct match, and a higher value indicates at what relative order the correct match was to be found. We find that the new method is more resilient to noise than previous methods tested.

Along with matching scores, we also wish to know to what degree each algorithm fails, if it does fail. To characterize each algorithm in these terms, we wish to compare values of metric distances from each test image to every one of the model images in terms of the distance from the test image to its corresponding correct model image. Therefore, for each test image, we divide these distances by the distance from the test image to the correct model image, and thus form distance ratios. Then a good characterization of performance is the mean value of distance ratios for *worst–case incorrect* models for each test (cf. [19]). Here, the worst case is the model that is closest to the test image, but is not the correct model. I.e., the worst case is the "next–best" mismatch model. Table 1 shows the mean and standard deviation over all experiments of this next–best mismatch distance, always using a metric distance. For the present database, this means that we give the mean and standard deviation of next–best distance ratios for the 65 incorrect model images over 30 tests. For algorithms that use an L2 distance metric, we use that distance. For algorithms that use histogram–match values μ with a maximum value of 1, we use the L1 distance $(1 - \mu)$. Comparing L1 distance with L2 distance is not entirely reasonable, but still distance ratio values do characterize algorithm performance. In Table 1 we show L1 distances in normal font and L2 distances in italics. Good performance is indicated by a mean distance ratio greater than 1 and the higher the better, since that would mean performance in which the algorithm rarely makes a wrong guess.

From the results in Table 1 we see that ChromHist–A performs best, and the ChromHist–B2 method, the version that operates entirely in the compressed domain, follows not too far behind. The ChromHist method in either the whole–histogram or compressed histogram version is least sensitive to noise. Below we show that for textures or when illumination change is actually present the ChromHist method in any form again does best of methods tested here. [3]

[3] Above we do not provide a comparison with the CCCI method [13]. That method selected only the 55 of Swain's 66 model images that had no pixel saturation. This concomitantly reduces the number of test images to 24. We do not preselect out any of

Table 1: Swain's images. Matches of 30 test images for database of 66 model images

Algorithm	Rankings				Next-Best Mismatch	
	1	2	3	> 3	Mean	SD
Color Indexing (L1)	23	2	0	5	1.32	0.49
Color Angles ($L2$)	24	1	2	3	*2.41*	*2.10*
ChromHist–A (L1)	28	2	0	0	1.47	0.36
ChromHist–B1 ($L2$)	26	1	1	2	*2.65*	*2.28*
ChromHist–B2 ($L2$)	27	1	1	1	*2.61*	*1.52*
With Added Noise						
Color Indexing (L1):	22	2	1	5	1.17	0.23
Color Angles ($L2$):	12	4	2	12	*1.02*	*0.71*
ChromHist–A (L1):	27	2	0	1	1.10	0.09
ChromHist–B1 ($L2$):	27	0	2	1	*5.24*	*5.77*
ChromHist–B2 ($L2$):	27	2	0	1	*4.59*	*4.47*

3.1.2 Lower–frequency images

Here we use methods ChromHist–C1 and –C2 outlined above in §4.2. For this and the other databases tested, we show results using an L2 norm because the L2 norm gave somewhat better performance than the L1 norm. Results are shown in Table 2, for number of diagonals in Fig. 2 from 1 to 4 (numbers of coefficients from 1 (== DC) to 10).

Table 2: Swain's images; 30 tests, 66 models— images stored in DCT form and DCT–inverted using `#DCT coeff's` DCT values.

	ChromHist–C1 #DCT coeff's					ChromHist–C2 #DCT coeff's				
Rankings	64	10	6	3	1	64	10	6	3	1
1	26	23	23	20	16	26	26	25	25	20
2	1	0	1	5	4	1	0	0	0	3
3	1	2	0	0	1	1	1	1	2	0
> 3	2	5	6	5	9	2	3	4	3	7

Somewhat surprisingly, method ChromHist–C2 performs betters than method ChromHist–C1. And in fact the former method does about as well as the compressed methods in Table 1, but using only a few DCT coefficients to restore the color image from its DCT format.

3.2 Textures: modeled illumination change

As another test, consider the model database of ten color texture images in [22]. For this model set, test images are created by modeling illumination change by means of imaging these textures through three separate colored filters. Thus there are a total of 30 test images under different simulated illuminations, for a model database of 10 images. After Finlayson et al. [3], we rotated test images by 30° with respect to the model image orientation before carrying out matching. As a result, the correlation-based method of [22] fails, with match rankings of 13 at rank 1, 7 at rank 2, 1 at rank 3, and 9 at rank > 3. A subsequent correlation-based method [23] addresses rotations, but we have not implemented that extension here.

In Table 3 results are shown for the six algorithms tested. Since it is asserted that image size is relatively unimportant for at least one of these methods [1], for the uncompressed methods we reduced the texture images

the images. However, we did perform runs using just these 55 model images, with no noise, using the ChromHist–A method and also our implementation of the Color Indexing method, for comparison with CCCI. For CCCI, results reported are: 23 matches of rank 1 and 2 of rank 2 out of 24 (*sic*). For this data, the Color Indexing method finds 21 matches with rank 1 and 3 matches with rank 2. Our method, ChromHist–A, finds 22 of rank 1 and 2 of rank 2.

by pixel averaging from 480×640 to 120×128.

Table 3: Textures. Matches of 30 reduced, rotated test images for database of 10 model images

Algorithm	Rankings				Next-Best Mismatch	
	1	2	3	> 3	Mean	SD
Correlation (*L2*)	13	7	1	9	*2.42*	*4.83*
Color Indexing (L1)	3	5	2	20	0.90	0.10
Color Angles (*L2*)	19	2	5	4	*1.23*	*0.61*
ChromHist–A (L1)	27	2	1	0	1.66	0.56
ChromHist–B1 (*L2*)	22	3	2	3	*3.29*	*4.73*
ChromHist–B2 (*L2*)	23	2	4	1	*2.33*	*1.75*

As can be seen, again methods ChromHist–B2 and B1 approximate fairly well to using full histograms and again the ChromHist method performs best. Note from the Next–Best Mismatch distance ratio that ChromHist–A performs much more robustly than its one L1–metric competitor; the mean of next–best mismatch values is 1.66 for ChromHist–A and 0.90 for Color Indexing. Thus the ChromHist method is much less likely to choose the wrong model image. [4]

For textures it does not make much sense to apply the blurred–image versions ChromHist–C1 and –C2; and in fact they do not perform as well as ChromHist–A or –B.

3.3 Objects: true illumination change

3.3.1 Full images

To test the indexing algorithms on data that truly reflects changing illumination and not just placing a filter in front of the camera lens, we consider the 13 model images in [3].

Test images consist of images of various objects, rotated, shifted, or wrinkled, taken under two different illuminants (three different illuminants in one case). Thus there are 27 test images for a model database of 13 images. Results for the algorithms tested are given in Table 4.

This database is the only one studied that truly utilizes changing illumination and here our method performs best, either in the whole–histogram or compressed–histogram version. As well, the new algorithm stands up best when challenged by noise.

3.3.2 Lower–frequency images

Again we apply methods ChromHist–C1 and –C2, based on lower–frequency versions of images restored from DCT format, with results as in Table 5.

Again we find that method ChromHist–C2 outperforms method –C1; and again we also see that we can do about as well reconstructing color images from just a few DCT components if they are already stored in that format as using full images.

4 Video Segmentation

Transition detection and characterization is an important aspect of digital video processing and many approaches have been used to contend with various confounding factors that make this seemingly simple task challenging. One salient theme has been the use of color or luminance histogram differencing for camera break ('cut') detection. Such methods for video segmentation rely upon the setting and tuning of thresholds for classifying interframe

[4] Of course, if test and model images were registered, then the very best one could do would be to model illumination change via a 3×3 transformation matrix among color bands; in [19] we computed a Least Squares approximation of such a matrix with excellent results. As well, we showed that wavelet–based compression followed by DCT and zonal coding did not destroy the linearity of the transformation and therefore that a Least Squares analysis could be carried out entirely in the compressed domain, leading to an indexing scheme with very little degradation from using full images and based on only 63 values.

Table 4: Objects under changing illuminants: Matches of 27 test images for database of 13 model images

Algorithm	Rankings				Next-Best Mismatch	
	1	2	3	> 3	Mean	SD
Color Indexing (L1)	2	6	7	12	0.76	0.35
Color Angles($L2$)	23	3	1	0	*3.48*	*3.64*
ChromHist–A (L1): uniform bins	24	1	2	0	1.24	0.25
ChromHist–A (L1): rand.image bins	23	2	2	0	1.20	0.26
ChromHist–A (L1): hist.-eq. bins	25	0	2	0	1.22	0.22
ChromHist–B1($L2$)	26	1	0	0	*5.00*	*5.57*
ChromHist–B2($L2$)	26	1	0	0	*3.50*	*2.29*
With Added Noise						
Color Indexing (L1):	3	3	1	20	0.56	0.36
Color Angles($L2$):	5	5	2	15	*0.71*	*0.55*
ChromHist–A (L1):	12	5	0	10	0.98	0.08
ChromHist–B1($L2$):	18	3	2	4	*2.09*	*2.32*
ChromHist–B2($L2$):	17	4	1	5	*1.83*	*1.73*

Table 5: Objects; 27 tests, 13 models— images stored in DCT form and DCT–inverted using `#DCT coeff's` DCT values.

	ChromHist–C1 #DCT coeff's					ChromHist–C2 #DCT coeff's				
Rankings	64	10	6	3	1	64	10	6	3	1
1	26	24	25	22	24	26	26	26	26	22
2	1	3	2	4	3	1	1	1	1	3
3	0	0	0	1	0	0	0	0	0	0
> 3	0	0	0	0	0	0	0	0	0	2

distances under various difference measures. A basic approach that has been used with some success has been to establish statistical measures for each new video and identify camera cuts as difference values far from the mean [24]. For this type of strategy the mean and dispersion for some interframe distance measure must be calculated for each new video as a whole. If we adopt the compressed chromaticity histogram for normalized channels as our image metric, then we can eliminate this statistical characterization step and at the same time allow for segmentation of streaming video by introducing a preprocessing step for illumination–invariance that concomitantly reduces input values to a uniform scale. The color constancy step provides a solution to the problem that simple changes of illumination in a scene, such as an actor emerging from a shadow, can trigger a false positive transition, no matter whether intensity alone or chrominance is used in a distance measure.

The illumination–independent video frame metric based on method ChromHist–B lends itself to a binary search approach to transition detection if we develop cut/no-cut thresholds for temporal intervals from 32 frames to 1 frame by powers of 2. To this end we examine distributions for intra–clip and inter–clip distances separately. Since video frames are all reduced to the same illuminant–invariant scaling, we can calculate such thresholds offline, for a training set of videos, and since therefore speed is not crucial here we use robust statistics to generate thresholds. Combining transition and non–transition distributions for each frame interval, we seek the valley between them as the threshold for each temporal interval.

Using the present method values of Precision and Recall are increased over previous methods. Moreover, illumination change produces very few false positives.

4.1 Methods Involving Non–Precomputed Thresholds

Ueda et al. [25] find shot boundaries using the change-rate of the color histogram. Arman et al. [26] first consider differences in Discrete Cosine Transformation (DCT) coefficients and then pass on candidate differences to a color hue–saturation histogram–difference thresholding step. Sethi and Patel [27] use histograms for the DC terms of each 8×8 block of I–frames for MPEG–encoded video and hypothesis testing for transition detection based on the AND of several more sophisticated statistics for the video under examination. Nagasaka and Tanaka [28] break each image into 16 regions and compute a χ^2 test on each, discarding the 8 largest differences.

However, as in many domains in computer vision, it is in the actual selection of a proper threshold that the difficulty lies for many video transition detection methods. Quick cuts are clearly the simplest transitions to detect, and yet other transitions such as fade–in, fade–out, and dissolves of various kinds may also involve threshold selection. Many of these measures use histogram differences, e.g. the L2 (Euclidean) or L1 (sum of absolute differences) norm. For example, in Zhang et al.'s method cuts are identified as histogram differences that lie beyond 5 or 6 standard deviations from the mean in any particular video; candidate special effects are thresholded with a much smaller threshold, marked, and then actual transitions are identified as those candidates leading to a cumulative difference over the larger, cut–detection, threshold.

Our approach avoids the computationally expensive necessity of statistically characterizing each new video, and at the same time establishes a hierarchical set of thresholds based on sound statistical grounds. Moreover, the metric used is (approximately) invariant to changes of lighting in a scene: for transitions for which other methods detect false cuts, our distance measure sees only a smooth variation — e.g., when an actor steps into a normal cast shadow on a sunny day our metric registers only a mild variation, whereas other methods see a sharp cut. As well, efficient video indexing of detected shots can be carried out based on only a 36–component vector [29].

We find here that using predefined thresholds of this type works well with the videos we have examined. Nevertheless, the statistical basis of the method could be used as a two–pass algorithm as well to adaptively set thresholds tailored to the input sequence.

4.2 Robust Thresholding and Binary Search

Since chromaticity values for normalized color channels are always in the range 0..1 for the images in any video, we examine the statistics of intra–clip (non–cut) and inter–clip histogram distances for an ensemble of typical videos and develop an overall threshold that approximately applies to any new video, without the need to statistically characterize the new video.

To effect an efficient search, we propose a cut detection scheme based on a binary search strategy. To that effect we wish to apprehend the difference between intra– and inter–clip chromaticity histogram differences over a series of frame intervals, *viz.* 32, 16, 8, 4, 2, and 1. Therefore we assemble a collection of typical intra–clip histogram differences and compare their statistics with those of a large collection of inter–clip (cut) distances. Together, these two kinds of distances make up a bimodal distribution, but one that is not necessarily simple to analyze. Therefore we begin by characterizing the mode of each separate kind of distribution using a mode–finder from robust statistics (see, e.g., [30]). Here we use the univariate version of the Least–Median–of–Squares (LMedS) method. Having identified the peak of the separate distributions for intra– and inter–clip histogram distances, for a particular frame interval (32 frames, say) we need a threshold at this frame interval to distinguish cuts from non–cuts. Combining the two distributions, then, we wish to identify the threshold between them using a valley–seeking mechanism. To do this, first we take the logarithm of all distance values since this has the effect of making each distribution steeper. Then inverting the distribution in the ordinate direction we can make use of the same LMedS mode–finder as used previously to identify the threshold.

Fig. 4 summarizes our two-phase method for cut-detection: (1) using a training data set to find the set of robust thresholds, (2) applying those (fixed) thresholds to any new video in a sequence of binary search.

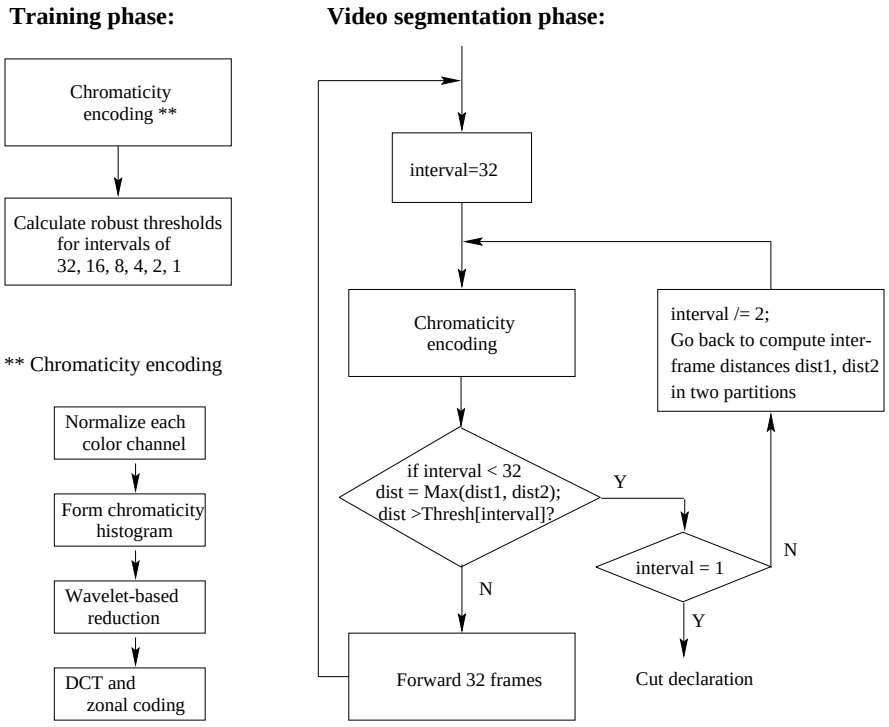


Figure 4: Calculation of thresholds, and subsequent binary search

While it is simple to collect many intra–clip distances from a set of videos, it is not so straightforward to assemble a sufficiently large set of inter–clip (i.e., clip transition) distances, since there are relatively few of them. Therefore we artificially create an ensemble of cut distances by randomly matching images from one clip with images from another, and calculating the distance between them.

Fig. 5(a) shows the distribution (as a probability) of distances for inter–clip histogram differences in an L2 metric, wherein the distance is the Euclidean distance between feature vectors. For a comparable set of intra–clip distances, the similar distribution is shown in Fig. 5(b). The median of cut distances is 31.6 and the range is 2.23 to 167. In comparison, the values for intra–clip (i.e., non–cut) distances have a much smaller median of 0.13, but the range crosses that for the cut distances: the range is 0 to 3.67. For larger frame intervals the overlap between cut and non–cut distances grows larger: e.g., for a frame interval of 32 frames, the maximum for intra–clip distances becomes 14.4.

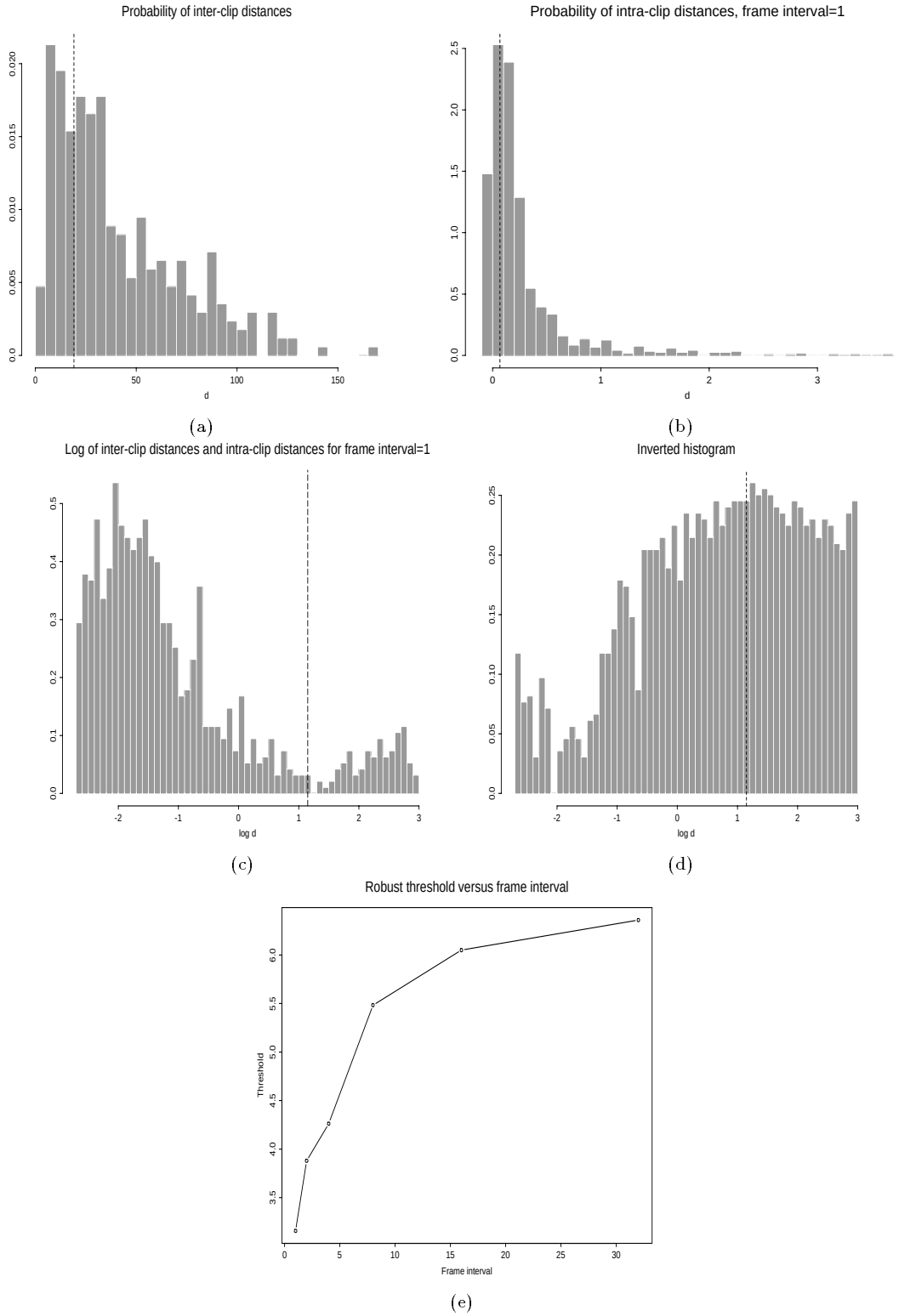


Figure 5: (a): Histogram of inter–clip (i.e., video transition) distances. The mode found by a robust estimator is shown as a vertical line. (b): Histogram of intra–clip (i.e., non–transition) distances, and the robust mode estimate. (c): Combined log of distributions; the robust threshold is the valley between the two modes. (d): Inverted histogram of log values. (e): Robust thresholds found, versus frame interval.

For video segmentation, we would like to find a threshold, for each temporal interval from 32 frames to 1 frame, to distinguish transitions from non–transitions. We can begin by looking for a concentration in the sample values, or in other words identifying the mode of each separate distribution, i.e., robustly characterize the *location* for each of the distributions in Figs. 5(a,b). Here we use the univariate version of the Least–Median–of–Squares (LMedS) method [30], which can effectively describe a distribution that is not simple while accurately discounting outlier values that could lead non–robust methods astray. For example, a mean minimizes the sum of squares, or in other words an L2 norm, and is subject to undue influence from a distribution with a long tail, such as the distances we encounter in a typical video sequence. The median, while more resistant to such outliers, is in principle equally flawed. In contrast, a robust method generally gives an excellent estimate provided at least 50% of values are inliers. The LMedS mode–finder proceeds by randomly selecting one of the possible values and calculating the median of the squared residuals from the chosen value. That data value with the smallest median in the residuals is identified as the mode estimate. In terms of a 1D estimator, then, we look for a mode estimate θ amongst distances d_i such that [30]

$$Min_\theta\{\mathrm{Med}_i((d_i - \theta)^2)\} \tag{16}$$

A robust estimate of the population dispersion is also provided

$$s_0 = k \left(1 + \frac{5}{n-1}\sqrt{\mathrm{Med}_i(r_i^2)}\right) \tag{17}$$

for n samples. Here the constant k is given in terms of the probability function Φ as $k = 1/\Phi^{-1}(0.75) \simeq 1.4826$, and r_i is the residual r for the i^{th} case, $r_i = d_i - \theta$. Then outliers are identified as values with residuals that are too large via

$$|r_i/s_0| > 2.5 \tag{18}$$

A final estimate of the mode is calculated by least sum–of–squares using only inlier values. Figs. 5(a,b) show the robust mode estimate by a vertical line.

If we simply concatenate the distributions in both types of distributions, cut and non–cut, the resulting distribution is not clear. However, if we first take the log of all distance values this has the effect of making each distribution steeper and the valley between them more evident. Fig. 5(c) shows both log distributions combined. The histogram is bimodal. To find the threshold separating the two modes we first truncate the histogram to distance values between the individual modes of the two kinds of distance. Then we would like to perform a valley–seeking procedure to find the threshold. Since we have already made use of a robust mode–finder, we can reuse that procedure if we first "invert" the distribution in the ordinate direction. To do so, we compose a new set of values that gives the same distribution as that shown in Fig. 5(c) — the robust mode finder works on sets of values as its input, not on a curve. First we subtract each bar in the histogram in Fig. 5(c) from the maximum bar height, effectively turning the distribution upside down. Then for each bar in the resulting distribution we place a number of data values in the new set equal to the bar height; a histogram of the new set of data values is the same as the inverted distribution of distances. This inverted distribution is shown in Fig. 5(d). Finally we can simply apply the robust LMedS method to the new data set, thus finding its mode, which is the valley in the non–inverted distribution. Figs. 5(c,d) show the threshold found at frame interval 1. Fig. 5(e) plots robust thresholds for frame intervals 1, 2, 4, 8, 16, and 32.

The hierarchical set of thresholds changes smoothly from higher values at a coarse frame interval to lower thresholds as the resolution increases. The threshold is roughly halved proceeding from an interval of 32 frames to an interval of 1 frame. The threshold changes because of overlap of the two distributions at higher frame intervals, for which intra–clip distances can be higher.

Based on such a hierarchical set we may carry out a binary search for video transitions by first examining inter–frame distances at a coarse temporal interval, say 32 frames or approximately 1 second interval, and then halving that interval to 16 frames if the distance measured is greater than the threshold for this level. If we continually move towards the lower inter–frame distance we may zero in on a video transition; alternatively if at any point in the search we discover that the distance has fallen below the threshold appropriate for that level of the hierarchy we may discontinue the search and move on in the video. Both alternatives lead to a considerably faster segmentation scheme than simply examining intervals of one frame. Below, in §5.3, we discuss the application of these hierarchical thresholds to the segmentation of videos that include more gradual transitions.

Note that it is only by the normalization and chromaticity steps that we are able to carry out a binary search based on precomputed thresholds.

Before discussing videos that are not in the training set, it is useful to determine just how well the robust thresholds perform on the training set itself in differentiating between transition and non–transition distances. Table 6 shows how many intra–clip distances (denoted $NumDists$) were used in generating the robust thresholds, for the various temporal intervals. The number of inter–clip distances was 338. As well, the number of actual cuts missed is shown as *Missed*. The number of intra–clip distances wrongly identified as cuts is shown as *FalsePositives*. Note that while values are shown for frame–intervals from 1 to 32, in actual practice the thresholds should be applied in the inverse order, from 32 to 1 frame intervals. The Recall and Precision statistics are also shown. Recall is the "percentage of desired items that are retrieved"; Precision is the "percentage of retrieved items that are desired items" [2]. Here, then, since cuts are the desired items, Recall is defined as `Recall = Correct/(Correct+Missed)` and Precision is defined as `Precision = Correct/(Correct+FalsePositives)`. As can be seen, for the training set itself both these values are very high for our method.

Table 6: Cut-detection statistics (out of 338 cuts) for the training set itself using hierarchical robust set of thresholds for compressed histograms.

	Frame interval					
	1	2	4	8	16	32
NumDists	1189	1163	1059	987	1121	1066
Missed	3	5	5	12	17	21
FalsePositives	3	0	2	3	10	38
Recall	0.99	0.99	0.99	0.96	0.95	0.94
Precision	0.99	1.0	0.99	0.99	0.97	0.89

5 Video Segmentation Results

5.1 Videos With Cuts

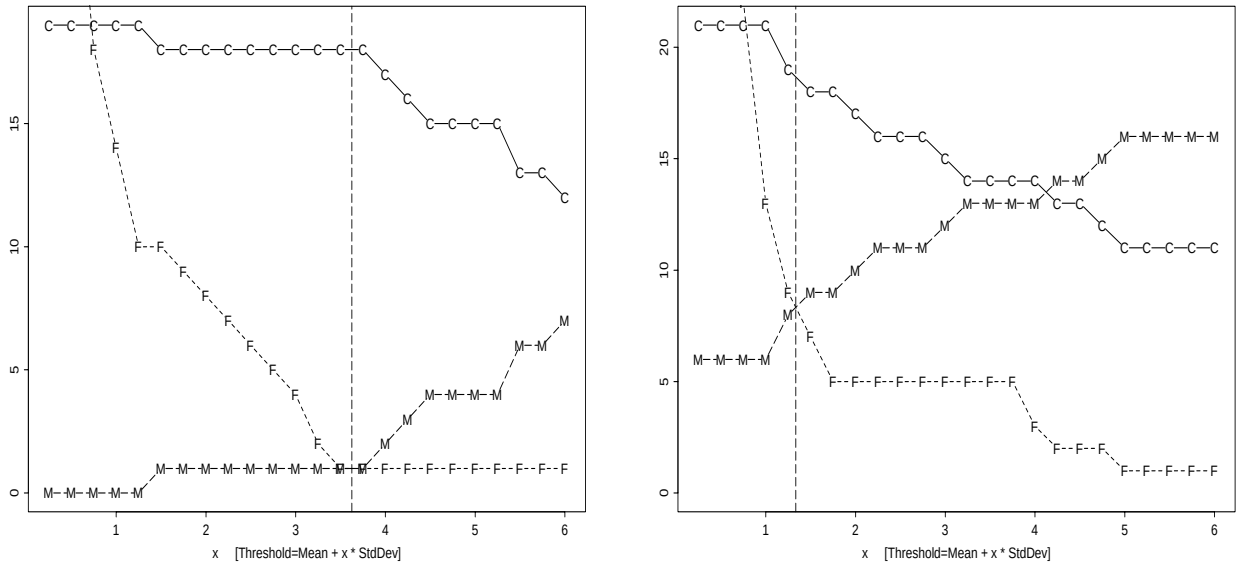
As an example of the efficacy of the method, consider the results shown in Table 7. Here we prepared a video with 1200 frames. This video consisted of 20 clips, with 19 cuts.

Table 7: Comparison of using gray value histograms, color histograms, and robust thresholds of chromaticity of color–channel-normalized images. The correct number of cuts is 19.

	Gray						Color						Normalization&Chromaticity					
Interval	1	2	4	8	16	32	1	2	4	8	16	32	1	2	4	8	16	32
Correct	14	14	16	16	13	8	16	16	17	17	16	15	18	18	17	17	17	15
Missed	5	5	3	3	6	11	3	3	2	2	3	4	1	1	2	2	2	4
FalsePositives	0	0	1	2	3	0	0	0	2	2	2	1	1	1	2	2	2	0

Here 'Gray' refers to using differences of gray value histograms, 'Color' refers to using differences of color histograms, and the third column shows results using robust thresholds for compressed histograms of chromaticity images formed from color–channel–normalized frames.

The present method, based on compressed chromaticity histograms of normalized color images, provides the best results and also has the advantage that no whole–video descriptive statistics for the new video need to be calculated. Nonetheless it would be interesting to see whether the color–based method, which provides the next–best results, could be tuned for this particular video (with 19 cuts) *using the a priori knowledge of where the cuts actually are*. To this end we formed thresholds of the form $T = mean + x * std.dev.$, for a set of multipliers x up to $x = 6$, for a 1–frame interval. This will allow us to investigate whether in fact a multiplier of 5 to 6 is



(a) (b)

Figure 6: Tuning color histogram based transition detection, using perfect knowledge of cuts.
Number of correct cuts `C`, of Missed cuts `M`, and FalsePositives `F` for the color–based method, over a range of thresholds `Threshold=Mean+x*StdDev` for $x = 0$ to 6, where `Mean` and `StdDev` are descriptive statistics for the whole video.
(a): Video 1 has 19 cuts. The best threshold is $Mean + 3.6 * StdDev$; the optimal value is shown as a vertical line.
(b): Video 2 has 22 cuts. Now the best threshold is $Mean + 1.3 * StdDev$.

actually appropriate for this video. The results, using color–based histogram differences, are shown in Fig. 6(a). Naturally, if the threshold is set very low then all the cuts are indeed found correctly — the number of correct cuts is shown by `C` in the plot. When the value of Correct is at a maximum, the number of Missed cuts, indicated by `M`, is low. But then the number of FalsePositives `F` is far too high. For a multiplier of 6, on the other hand, Correct is too low (and therefore the Recall statistic is too low) and Missed is too high. The Precision climbs to 100% for a high threshold, but that is not useful since it merely reflects the high threshold. Nevertheless, with perfect knowledge we are able to tune the threshold by setting the multipler to $x = 3.6$. [5]

However, for another video, consisting of 22 clips and 2016 frames, this tuning parameter is quite different: Fig. 6(b) shows that the optimal tuning parameter in this case is about $x = 1.3$, where we identify optimality somewhat empirically with the crossing of the Missed and FalsePositive lines.

Table 8: Another clip, with 2016 frames.The correct number of cuts is 21.

	Gray						Color						Normalization&Chromaticity					
Interval	1	2	4	8	16	32	1	2	4	8	16	32	1	2	4	8	16	32
Correct	13	18	14	17	17	16	17	18	14	18	20	19	20	19	17	17	16	13
Missed	9	3	7	4	4	5	4	3	7	3	1	2	1	2	4	4	5	8
FalsePositives	4	3	12	5	4	1	3	1	9	18	13	1	3	1	0	0	0	0

Therefore even with perfect knowledge the color–based method cannot easily be optimized by simply setting a fixed value for x.

Table 8 shows the results of the three methods of cut detection tested for this second video.

5.2 Videos with Illumination Changes

(a) (b) (c) (d)

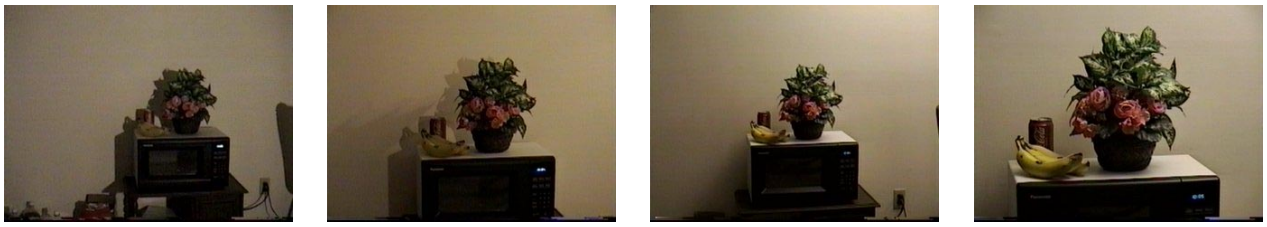
Figure 7: A child steps out of a shadow on a sunny day.

Since the present method is meant to discount illumination changes, whereas other methods do not take into account such variations, we examined continuous video sequences containing changes in illumination but no straight cuts. We found that previous methods miscategorized as cuts all illumination changes in these scenes. In contrast, our method classified these lighting shifts as smooth.

Fig. 7 shows some stills from a 1 second video taken as a child steps out of a cast shadow. Here, the Recall statistic does not characterize cuts, since there are none. Instead, it may be used to describe how many intra–clip distances were correctly classified as non–cuts. [6]

[5] Note that in fact `C` and `M` always add to the number of cuts so that one of them is redundant; nevertheless it is interesting to visualize how they mutually change and the figure includes both for that reason.

[6] The Precision statistic does not apply in this case since we are inverting the usual definition of success, *viz.* how many cuts were found; instead, we are interested in how many non–cuts were found. Thus there are no "false positives", which would be cuts incorrectly classified as non–cuts, since there are no cuts.



(a) (b) (c) (d)

Figure 8: A video with lighting changes.

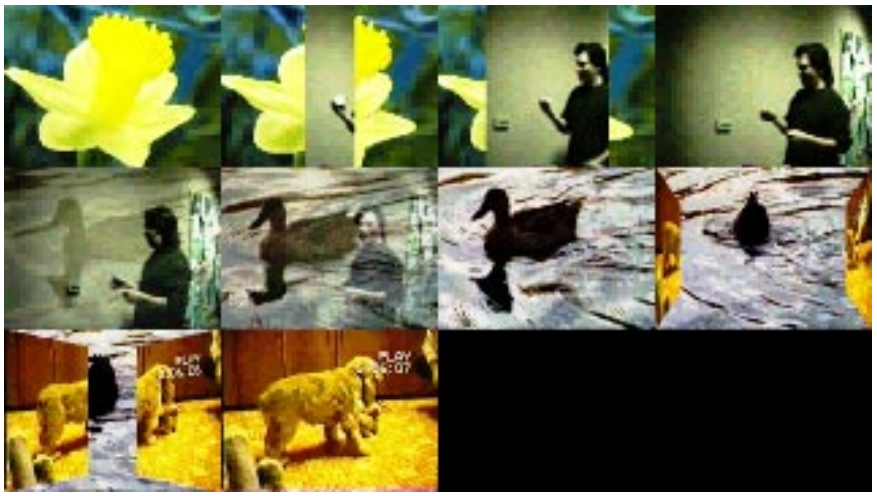


Figure 9: A video with gradual and special–effect transitions.

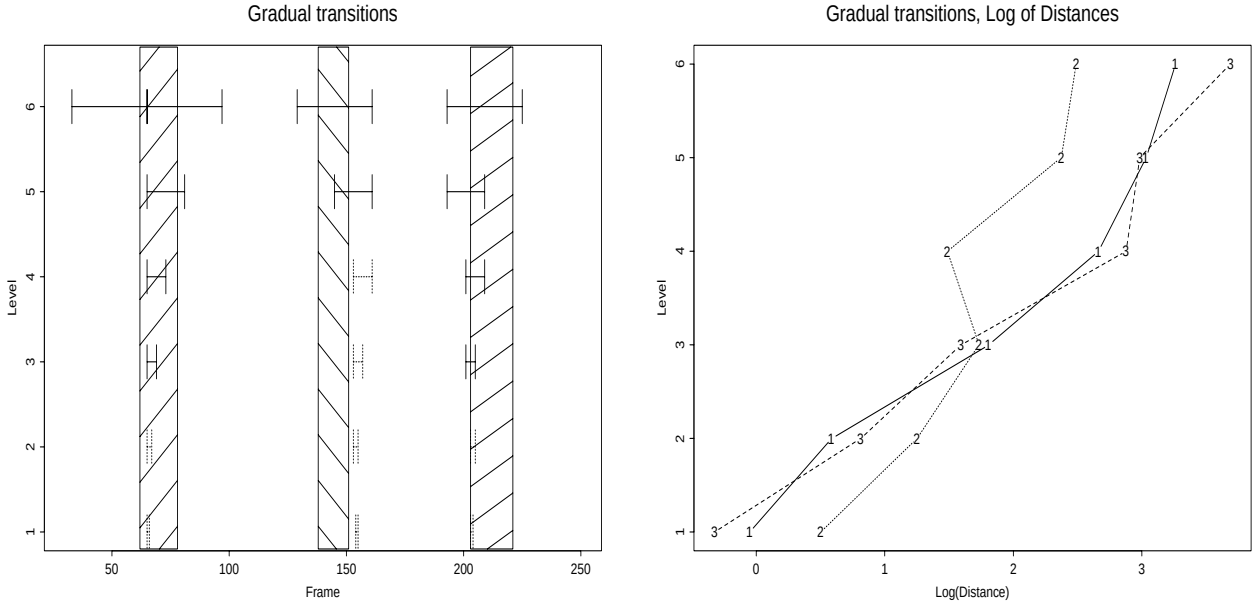
Our method had 100% Recall at all frame intervals. Other methods we tested had from 50% Recall at frame interval 16 to 83% Recall at frame interval 1 using grayscale histograms, and from 36% to 90% Recall using color–based histograms at the decreasing temporal interval sizes, with the illumination–change clip embedded in our training set of video clips. Therefore, not only does the new method fare better but also avoids miscategorizing illumination changes as scene transitions.

In Fig. 8, a series of frames from another video clip that includes lighting changes is shown. This clip consists of 202 frames, with no cuts. It starts with a room lamp illuminating the scene from the front–right, then the lighting is changed by adding another light source, a desk lamp, at about the same position. The third image is a representative frame after a change to very strong frontal lighting, and the clip finishes with a zoom in. The images shown are typical frames within the three different illumination conditions. Most of the color change comes about because of change in illumination.

For previous methods, either gray–based or color–based, the two lighting changes are incorrectly identified as cuts. For the chromaticity–based method, the correct result, zero cuts, is found.

5.3 Videos with Gradual and Special–Effect Transitions

This subsection shows that our cut-detection method can also be extended to detect smoother transitions. Consider a video with several "special effects" transitions — a central wipe, an additive dissolve (i.e., fade–in plus fade–out), and a "barn doors" wipe. Fig. 9 shows representative frames from a 19 second video with these three transitions, sampled at 15 frames per second. An initial screening using the pre–determined robust thresholds for an interval of 32 frames produces some candidate transitions at Level 6 depicted with horizontal lines in Fig. 10(a). The vertical crosshatched regions show the actual extent of the three transitions in this video. At this level, there are two segments identified as above the threshold, for the first transition. These are merged in the next level, Level 5 (or interval=16 frames).



(a) (b)

Figure 10: (a): Binary search. Gradual transition regions are indicated by crosshatched bars. Horizontal lines at Level 6 (32 frame interval) are above the threshold at that level. Distances examined at lower levels are shown dotted once they fall below the threshold at any level. (b): Distances for three types of gradual special–effect transitions, starting from transitions detected at frame interval 32 (Level 6) and halving frame intervals down to Level 1, at the lowest resolution of a 1–frame interval.

A binary search that always goes to the higher value of inter–frame distance at any level of frame interval

produces horizontal bars as in Fig. 10(a) and these bars are halved in length at each successive level of the hierarchy from Level 6 to Level 1. We display these horizontal bars as solid lines if the distance is above the threshold at the current temporal interval, and dotted if below the threshold. As expected, eventually distances at the finer resolutions go below the thresholds since in fact the transitions are gradual, not sharp cuts. Our program hence declares the detection of all three transitions.

In comparison, we implemented the "twin–threshold" method of Zhang et al. [24] Here, possible gradual transitions are first identified using a smaller threshold for consecutive frames, and then a gradual transition is identified if the cumulative distance goes above their larger, cut–identifying, threshold. For this video, their method did not succeed. Using grayscale histograms, the twin–threshold method identified 11 gradual transitions in this clip, whereas in actuality there are only 3. For color–based histograms, 13 gradual transitions were signaled.

Fig. 10(b) shows the distances found by the binary search from Level 6 (32 frame interval) to Level 1 (single frame interval). The three transitions are labeled `1,2,3`, and distances are shown on a log scale. It is interesting to note that the dissolve, which is transition `2`, has quite a different pattern than the other two transitions, which are each a type of wipe. In particular, the dissolve is not monotonic, whereas the others are. We intend to study such characteristics in greater detail in order to develop a taxonomy or set of signatures for how various transition types behave in this type of plot.

6 Conclusion

The simple idea of normalizing color images separately in each band is a reasonable approach to color constancy preprocessing in the tasks addressed here. The method of using histograms of chromaticity and applying a wavelet–based image reduction transformation for low–pass filtering along with DCT and truncation results in an image feature vector with only 36 or 72 values. The algorithm is a careful combination of simplicity along with complexity where it is useful. Results for experiments on test images with no change of illumination, illumination change modeled by changing camera filters, and true change of illumination, with and without noise added, show that the method does better or a good deal better than the other methods tested. We find that operating entirely in the compressed–histogram domain has an appreciable but acceptable effect on the accuracy of the method, compared to using whole images and no compression.

As well, if images are already stored in the DCT domain then they can be normalized within that domain and then DCT–inverted with only a few DCT coefficients, for a very simple low–pass filtering of the image. Somewhat surprisingly, the best results using DCT–encoded images came from first DCT–inverting using a few coefficients, and then normalizing, for the image databases examined.

The fact that the color–channel normalization step reduces every image to the same scale means that the present method can also be applied to transition detection in video. Because transition thresholds can be predetermined from a training set of videos, new videos or streaming videos can be segmented without having to generate descriptive statistics for the entire video, for each new video. The metric used in this domain is again based on compressed chromaticity histograms. Here we developed a method for obtaining a hierarchical set of video–independent robust thresholds for detection of video transitions. The advantage of a binary search method with possibly variable thresholds is that a candidate cut can be tentatively classified at a coarse resolution and then later be correctly declared merely an admissible intra–clip change at a finer resolution — it is not an all–or–nothing decision.

The robust thresholds also apply to the detection of non–sharp transitions and special effects; since we have available hierarchical information, potential cut candidates might be further classified using the behavior of values during the binary search.

Since the metric used for image differences is based on histograms, 2D changes of object pose or position do not change distances found. As well, since the metric is approximately color constant, illumination changes that are miscategorized as cuts in other methods are correctly seen as smooth changes. We found that the present method performs as well or better than other methods tested on straight cuts, in terms of both Recall and Precision, and performs a good deal better on clips that include lighting changes.

Further work remains to be done on the correct classification of gradual and special–effect transitions, using the new metric. In particular, it would be interesting to robustly characterize the statistics of different kinds of transitions by the way in which histogram–difference distances decrease as the resolution becomes finer.

References

[1] M.J. Swain and D.H. Ballard. Color indexing. *Int. J. Comput. Vision*, 7(1):11–32, 1991.

[2] J.S. Boreczky and L.A. Rowe. Comparison of video shot boundary detection techniques. In *Storage and retrieval for image and video databases IV*, pages 170–179. SPIE Vol. 2670, 1996.

[3] G.D. Finlayson, S.S. Chatterjee, and B.V. Funt. Color angular indexing. In *ECCV96*, pages II:16–27, 1996.

[4] M. Yeung and B. Liu. Efficient matching and clustering of video shots. In *International Conference on Image Processing*, pages Vol. 1, 338–341, 1995.

[5] W. Y. Ma and B. S. Manjunath. NeTra: A toolbox for navigating large image databases. In *Int. Conf. on Image Proc.*, page I:568. IEEE, 1997.

[6] G.D. Finlayson, M.S. Drew, and B.V. Funt. Spectral sharpening: sensor transformations for improved color constancy. *J. Opt. Soc. Am. A*, 11(5):1553–1563, May 1994.

[7] C.F. Borges. Trichromatic approximation method for surface illumination. *J. Opt. Soc. Am. A*, 8:1319–1323, 1991.

[8] G. Healey and A. Jain. Retrieving multispectral satellite images using physics–based invariant representations. *IEEE PAMI*, 18:842–848, 1996.

[9] G.D. Finlayson, M.S. Drew, and B.V. Funt. Color constancy: diagonal transforms suffice. *J. Opt. Soc. Am. A*, 11(11):3011–3019, Nov. 1994. Special issue on physics–based vision.

[10] M.S. Drew, J. Wei, and Z.-N. Li. Illumination–invariant color object recognition via compressed chromaticity histograms of color–channel–normalized images. In *ICCV98*, pages 533–540. IEEE, 1998.

[11] G. Wyszecki and W.S. Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulas*. Wiley, New York, 2nd edition, 1982.

[12] G. Healey and L. Wang. The illumination–invariant recognition of color. In *ICCV95*, volume 12, pages 128–133, 1995.

[13] B.V. Funt and G.D. Finlayson. Color constant color indexing. *IEEE PAMI*, 17:522–529, 1995.

[14] D. Slater and G. Healey. Combining color and geometric information for the illumination–invariant recognition of 3–d objects. In *ICCV95*, pages 563–568, 1995.

[15] D. Berwick and S.W. Lee. A chromaticity space for specularity, illumination color- and illumination pose-invariant 3-D object recognition. In *ICCV98*, pages 165–170. IEEE, 1998.

[16] K.D. Skifstad and R.C. Jain. Illumination independent change detection for real world image sequences. *CVGIP*, 46:387–399, 1989.

[17] G. Healey. Using color for geometry-insensitive segmentation. *J. Opt. Soc. Am. A*, 6:920–937, 1989.

[18] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. Image coding using wavelet transform. *IEEE Trans. on Image Processing*, 1(2):205–221, 1992.

[19] M.S. Drew, J. Wei, and Z.-N. Li. On illumination invariance in color object recognition. *Pattern Recognition*, 31:1077–1087, 1998. Also SFU-CMPT Tech. Rep. TR 97-07, ftp://fas.sfu.ca/pub/cs/TR/1997/CMPT97-07.ps.gz.

[20] Mark S. Drew, Jie Wei, and Ze-Nian Li. Illumination–invariant color object recognition via compressed chromaticity histograms of normalized images. Technical Report TR 97-09, Simon Fraser University, April 1997. ftp://fas.sfu.ca /pub /cs /TR /1997 /CMPT97-09.ps.gz.

[21] B.L. Yeo and B. Liu. A unified approach to temporal segmentation of motion JPEG and MPEG compressed video. In *IEEE Int. Conf. on Multimedia Computing and Sys.*, pages 81–88, 1995.

[22] G. Healey and L. Wang. Illumination–invariant recognition of texture in color images. *J. Opt. Soc. Am. A*, 12:1877–1883, 1995.

[23] L. Wang and G. Healey. Illumination and geometry invariant recognition of texture in color images. In *CVPR96*, pages 419–424, 1996.

[24] H.J. Zhang, A. Kankanhalli, and S.S. Smoliar. Automatic partitioning of full-motion video. *Multimedia Systems*, 1:10–28, 1993.

[25] H. Ueda, T. Miyatake, and S. Yoshizawa. IMPACT: An interactive natural-motion-picture dedicated multimedia authoring. In *Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems*, pages 343–350, 1991.

[26] F. Arman, A. Hsu, and M.-Y. Chiu. Feature management for large video databases. In *Storage and retrieval for image and video databases*, pages 2–12. SPIE Vol. 1908, 1993.

[27] I.K. Sethi and N. Patel. A statistical approach to scene change detection. In *Storage and retrieval for image and video databases III*, pages 329–339. SPIE Vol. 2420, 1995.

[28] A. Nagasaka and Y. Tanaka. Automatic video indexing and full-video search for objectappearances. In L. Wegner E. Knuth, editor, *Visual Database Systems II*. Elsevier Science Publishers, 1992.

[29] J. Wei, M.S. Drew, and Z.-N. Li. Illumination invariant video segmentation by hierarchical robust thresholding. In *Electronic Imaging '98: Storage and Retrieval for Image and Video Databases VI*, pages 188–201. SPIE Vol. 3312, 1998.

[30] P. J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection*. Wiley, 1987.