A Color-Based Classifier for Region Identification in Video

Richard P. Schumeyer Kenneth E. Barner
Department of Electrical and Computer Engineering
University of Delaware
Newark, DE 19716-3130 USA

ABSTRACT

Content based coding has been proposed by several authors and members of the MPEG-4 community as a solution to very low rate video coding. Using content coding, a video sequence is decomposed into objects that may be encoded independently. Such a scheme requires a fast and accurate segmentation algorithm to identify various objects in the scene. In this paper we propose, develop, and analyze a color-based segmentation algorithm.

One application of interest is coding of sign language video sequences. The requirements for accurate perception of sign language differ from those of traditional head-and-shoulders videoconferencing sequences. We propose a content-based coding method in which perceptually important regions in an image are identified, and more resources are allocated to these regions. Since face, hands and arms are important components of sign language, regions are defined that encompass these features.

The dynamic segmentation algorithm identifies flesh regions using statistical methods operating on image color distributions. A method for performing the segmentation in the perceptually linear $L^*a^*b^*$ space using data captured in the YC_bC_r space is developed. Results of encoding sign language sequences using the proposed content-based methods illustrate the improved quality that can be achieved at a the same bit rate when compared to a uniform algorithm.

Keywords: content-based coding, color segmentation

1. INTRODUCTION

Video coding for transmission over very low rate channels is receiving considerable attention. Coding for very low rate channels requires novel compression schemes. Content-based coding offers a more flexible and intelligent approach than traditional methods embodied by standards such as MPEG-2 and H.263. The emerging MPEG-4 standard will include some form of content-based functionality; several proposals containing content-based ideas were recently published.[1–3] Using content coding, video sequences are typically segmented into regions which may then be independently coded and transmitted. An algorithm to segment the video stream into different objects is a key element of such a scheme. In this paper, we analyze and develop a color-based classifier to perform the region segmentation.

The proposed classifier was designed to be effective for segmentation of sign language sequences. Video transmission over affordable low rate channels will allow hearing impaired individuals to communicate remotely via sign language, giving this population the same advantages that the telephone provides for hearing individuals. Most videoconferencing research to date, however, has focused on head-and-shoulders sequences, which have characteristics different than that of sign language sequences.[4]

The developed video coder improves the quality of perceptually important image regions, relative to traditional coding standards, at the expense of perceptually insignificant portions of the image. The first step is to identify the region-of-interest (ROI). Since face, hands and arms are important components of sign language, the proposed method defines a region that encompasses these features. Color distributions are modeled as Gaussian mixtures, and

(Send correspondence to K.E.B)
K.E.B.: Email: barner@ece.udel.edu; Telephone:302-831-6937; Fax: 302-831-4316;
R.P.S.: Email: schumeye@asel.udel.edu
Supported by the National Science Foundation under grant #HRD-9450019.

individual pixels are Bayes classified as either "flesh" or "background". The result of the pixel level classification is post-processed to determine the macroblocks in the ROI.[5] This paper discusses several issues related to the design of the color-based classifier, including the use of color as a segmentation feature, the choice of colorspace, and color look-up table (LUT) configuration.

The remainder of the paper is organized as follows. Section 2 contains a thorough analysis of color related design issues. Section 3 describes the configuration of the video coder. The performance of the segmentation algorithm on sequences is shown in Section 4, and conclusions are presented in Section 5.

2. Color-Based Flesh Segmentation

2.1. Color as a feature for segmentation

The goal of the segmentation algorithm is to identify image pixels corresponding to the hands and face. There are several examples of face segmentation methods, and they can be roughly classified into two groups: feature extraction and template matching.[6] Typically in the feature extraction case, high level features, such as mouth position, are derived from low level features, such as edge location. The template matching approach often uses raw image data. The survey given in[7] describes many face finding algorithms. The flesh segmentation method presented here uses only color, which is a low level feature.

Many feature extraction methods are based on edge detection.[8–10] Such methods require careful control of lighting and background. For example, shadows can cause "extra" edges in the middle of the face, presenting an additional challenge to the algorithm. Additionally, a segmentation method for sign language must also find hands. This can be difficult using edges since the edge properties of the hand depend on orientation and finger position.

Template matching is another approach to feature location that has been used by several authors.[6,11] This method can effectively locate faces, since all faces have a similar structure. The use of templates to find hands has the same difficulties as using edges, namely that the hand shape varies widely depending on wrist and finger position.

Motion information is another choice for segmentation of video sequences.[1,12] This approach utilizes the idea that distinct objects move differently. The motion at many points in the image is measured, and similar motion measurements are grouped together into objects. The problems associated with hand orientation are eliminated. If motion vectors are already being calculated for motion compensation, then the computation cost associated with this method is greatly reduced.

Color based segmentation is an alternative approach with several advantages over competing methods. Color methods do not suffer from the difficulties affecting edge and template methods: the face and hand colors are similar for a given person, and the color varies little with changes in hand orientation or finger position. Speed favors the use of color as well. Using color as a feature is generally faster than computing edge based features or template cost functions for every frame. Using color, a decision can be made for each pixel. In contrast, motion estimates are usually made on a block basis.

While there are numerous benefits associated with the use of color, some disadvantages exist. For instance, many objects commonly found in backgrounds, such as wood doors, are similar in color to flesh tones, making the segmentation task more difficult. Further, color does not take advantage of spatial correlation, unless it is combined with other techniques. Despite these drawbacks, color will be shown to provide fast, effective segmentation.

Although most face extraction algorithms are not color based, the literature contains several examples of color-based segmentation algorithms. Duchnowski *et al.*[13] used normalized RGB values in their face color classifier. This method uses several training images to form an initial face color distribution; therefore no new training was required for new images. After locating all flesh colored regions in the image, the shape of these regions was examined to eliminate other regions, such as hands. Similarly, Sobottka and Pitas[14] declared *a priori* a region of the H-S plane of the HSV space to contain skin colors. The skin color region corresponding to the face was also determined during post-processing by fitting an ellipse to each skin region. The eyes and mouth were then found by searching for dark areas within the face. They claimed an 86% facial feature detection rate.

In another approach, Lee[15] used motion information to locate a moving face in a video sequence. A velocity vector field was calculated and thresholded, yielding the region containing the face. Color information was then used to find eyes, eyebrows, and mouth. Lee found that the HSI space provided the greatest separation of skin, eyes, and mouth.

Dai and Nakano[16] converted RGB images to YIQ, and used the I component as their feature. They used a space gray-level dependence matrix (SGLD) to provide a facial texture model. The SGLD matrix used only the I component. They reported an identification rate of 98% on the ORL database.

2.2. Choice of Colorspace

The previous references show that many choices of color space are available and have been used for image analysis. The cited authors neither discussed the choice of color space nor presented objective comparisons of various spaces. This section considers several color spaces and discusses the advantages of the $L^*a^*b^*$ space.

Although images are displayed in the RGB space, analysis is not often conducted in this space since the components are highly correlated. Instead, researchers use spaces such as NRGB, YIQ, HSV, $L^*a^*b^*$ or YC_bC_r. Transformations from RGB to these other spaces are further discussed by Foley[17] and Jain.[18]

The design of the $L^*a^*b^*$ space provides certain advantages for segmentation. The $L^*a^*b^*$ space was experimentally designed to be perceptually linear. This is a valuable property for machine vision, since humans are very good at segmenting objects based on color. To illustrate the advantages of the $L^*a^*b^*$ space for flesh segmentation, a composite face color distribution drawn from three faces containing different skin tones was transformed from RGB to YC_bC_r, $L^*a^*b^*$, and HSV. The resulting distributions are shown in Figure 1, and the normalized eigenvalues of the covariance matrices,

$$\mu_i = \frac{\lambda_i}{\sum_{j=1}^{n} \lambda_j} = \frac{\lambda_i}{\operatorname{tr} \mathbf{C}_X} \tag{1}$$

are shown in Table 1. The flesh distributions in both the RGB and YC_bC_r space are dominated by one eigenvalue, indicating that the color distributions have large variance in that dimension. There is less spread in the eigenvalues in the $L^*a^*b^*$ and HSV spaces, indicating that the flesh points are more uniformly distributed in these spaces. Note that the distributions are more compact in the $L^*a^*b^*$ and HSV spaces. This is expected, since the flesh colors present in the distribution, while different, are perceptually similar. Since $L^*a^*b^*$ is perceptually linear, these colors fit in a more uniform region. The color based segmentation is thus performed in the $L^*a^*b^*$ space since the flesh colors occupy a compact and distinct region of that space.

Although the $L^*a^*b^*$ space is three-dimensional, the dimensions do not contribute equally to discrimination. The normalized eigenvalues of the covariance matrix in Table 1 reveal that there is significantly more variation in the L^* dimension. This suggests that L^* may be less relevant than the other components for discrimination between flesh and non-flesh colors. To illustrate this, 250 Gaussian test clusters were uniformly distributed throughout the space. These test clusters represent the range of possible background colors. The scatter between the flesh cluster and the test clusters was measured, both in the 3D $L^*a^*b^*$ and in the 2D subspaces L^*a^*, L^*b^*, and a^*b^*. If the scatter measure in one of the subspaces is only slightly less than the scatter of the full space, then segmentation may be effectively performed in that subspace since it provides nearly the same discrimination ability. Additionally, using only two dimensions reduces the number of parameters to be estimated, which is important since only a limited number of training samples are usually available.

Scatter matrices are often used to measure the ability to discriminate between clusters.[19] Let ω_i represent class i, $\boldsymbol{\mu}_i$ be the mean of the samples in class i, and $\mathbf{x}$ be a sample in the color space. Then for the case of N classes, the within-class scatter is

$$S_w = \sum_{i=1}^{N} p(\omega_i) E\{(\mathbf{x} - \boldsymbol{\mu}_i)(\mathbf{x} - \boldsymbol{\mu}_i)^T\} = \sum_{i=1}^{N} p(\omega_i) \mathbf{C}_i, \tag{2}$$

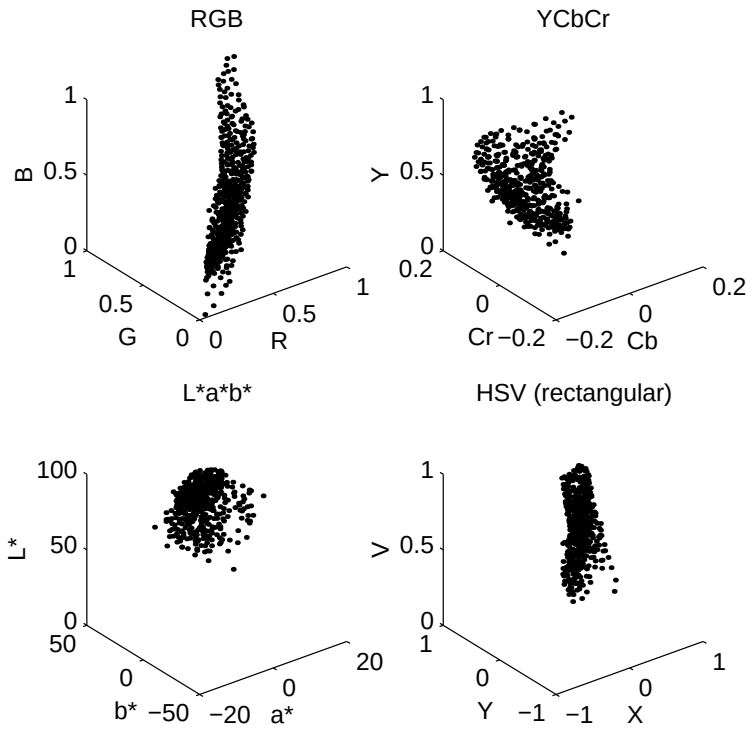

Figure 1: Distribution of face colors in different color spaces.

$L^*a^*b^*$	RGB	YC_bC_r	HSV
Normalized Eigenvalues			
0.5897	0.9388	0.9355	0.3413
0.3189	0.0534	0.0558	0.5788
0.0914	0.0078	0.0088	0.0849
1^{st} Principal Component			
$\begin{bmatrix} 0.98 \\ -0.12 \\ -0.18 \end{bmatrix}$	$\begin{bmatrix} 0.61 \\ 0.61 \\ 0.51 \end{bmatrix}$	$\begin{bmatrix} 1.00 \\ -0.10 \\ 0.02 \end{bmatrix}$	$\begin{bmatrix} 0.94 \\ -0.29 \\ 0.19 \end{bmatrix}$

Table 1: Normalized eigenvalues of covariance matrices of face color distributions in $L^*a^*b^*$, RGB, YC_bC_r, and HSV, along with eigenvector associated with the 1^{st} principal component. For HSV, calculations were done in rectangular coordinates, so $X = S\cos H$, $Y = S\sin H$.

and the between class scatter is given by

$$S_b = \sum_{i=1}^{N} p(\omega_i) E\{(\boldsymbol{\mu}_i - \boldsymbol{\mu}_0)(\boldsymbol{\mu}_i - \boldsymbol{\mu}_0)^T\}, \tag{3}$$

where $\boldsymbol{\mu}_0 = E\{\mathbf{x}\} = \sum_{i=1}^{N} p(\omega_i)\boldsymbol{\mu}_i$. The scatter criterion is defined as $J = \mathrm{tr}(S_w^{-1}S_b)$. The eigenvalues of $S_w^{-1}S_b$ measure the ratio of between-cluster to within-cluster scatter in the direction of the eigenvectors.[20] Since J is the sum of these eigenvalues, larger values of J imply better discrimination ability.

Measuring the scatter of all test background clusters and the face cluster together would measure not only the separability between the face and test clusters, but also between each test cluster. We are only concerned with

Subspace	$\bar{S}_i$	var S_i
L^*a^*	0.52	0.10
a^*b^*	0.86	0.03
L^*b^*	0.61	0.10

Table 2: Mean and variance of retained scatter of the 2D subspaces compared to the 3D space.

the scatter between flesh colors and the portion of background represented by each test cluster. Thus, the scatter criterion of each test cluster alone with the flesh cluster was measured in the full 3D space as well as the 2D subspaces L^*a^*, L^*b^* and a^*b^*. The retained scatter, S, is defined as the ratio of the 2D scatter versus the 3D scatter:

$$S_i = \frac{J_{2i}}{J_{[\mathrm{Lab}]i}} \qquad i \in [1...250], \tag{4}$$

where J_2 may be any of the 2D scatter measures $J_{[\mathrm{La}]i}$, $J_{[\mathrm{Lb}]i}$ or $J_{[\mathrm{ab}]i}$. The mean and variance of the retained scatter for each subspace is shown in Table 2. These results confirm that the a^*b^* subspace retains most of the discrimination ability of the 3D space.

2.3. Mapping from YC_bC_r to a^*b^*

Although it is advantageous to perform segmentation in the a^*b^* space, video hardware typically captures images in YC_bC_r or YIQ. The transformation from YC_bC_r to a^*b^* is nonlinear and computationally expensive; it is prohibitive to transform the pixels in each frame. Video hardware typically has a resolution of 8 bits in each of the Y, C_b, and C_r components, resulting in 2^{24} possible observation values. Instead of transforming each image pixel, each possible YC_bC_r value could be transformed to a^*b^* for segmentation, and the classification result stored in a look-up-table (LUT). Segmentation would then consist of referencing a cell in the LUT according to a pixel's YC_bC_r value. Such a LUT, however, would contain 2^{24} cells and can be an inefficient use of available memory. To limit memory use, we propose a method in which only a subset of the possible YC_bC_r values are transformed to a^*b^* and then classified. If the subset is chosen properly, a LUT much smaller than the full LUT can be used without introducing significant distortion. The remainder of this subsection discusses the optimization of such a LUT.

Let N_y, N_b, and N_r be the resolution in bits in the LUT for components Y, C_b, and C_r, respectively. Then the sum $N_L = N_y + N_b + N_r$ is the size of the LUT. The *shape* of a LUT refers to the apportionment of the N_L bits among N_y, N_b, and N_r. If the hardware resolution is 8 bits per component, then the maximum value of N_L is 24. Many LUT shapes are possible for values of N_L less than 24. For a given value of N_L, we wish to find the LUT shape that introduces the least distortion. For the purpose of optimizing LUT shape, distortion occurs if a color is classified differently than with the maximum size LUT.

Restrictions are placed on the mapping from the full space to the LUT in order to ease the optimization. Thus, N_y, N_b, and N_r are each restricted to be power of 2, and the mapping of each component is constrained to be uniform. This mapping is achieved by extracting the most significant bits of each component. For example, if $N_b = 6$ then right-shifting the C_b value by two bits leaves the most significant 6 bits. This shifted value is the index into the table for that component.

Let $\mathbf{x}$ be a point in YC_bC_r, and let $S_y = 256/2^{N_y}$ be the number of Y values that map to one LUT index. Define S_b and S_r similarly for C_b and C_r. Then the quantized value of $\mathbf{x}$ is given by

$$\hat{\mathbf{x}} = \left[S_y \left(\left\lfloor \frac{Y}{S_y} \right\rfloor + \frac{1}{2} \right), \quad S_b \left(\left\lfloor \frac{C_b}{S_b} \right\rfloor + \frac{1}{2} \right), \quad S_r \left(\left\lfloor \frac{C_r}{S_r} \right\rfloor + \frac{1}{2} \right) \right]^T. \tag{5}$$

If we assume that the segmentation resulting from the full LUT is the best possible classification, then we can define distortion as a classification different from that of the full LUT:

$$D_L(\mathbf{x}_i, \hat{\mathbf{x}}_i) = |F(\mathbf{x}_i) - F(\hat{\mathbf{x}}_i)|, \tag{6}$$

where $F(\mathbf{x}_i)$ is the output of a Bayes classifier. $F(\mathbf{x}_i)$ represents the class to which a pixel belongs, based on its' color. The goal of the optimization is to minimize the expected distortion $E[D_L(\mathbf{x}_i, \hat{\mathbf{x}}_i)]$. In practice, the optimal

N_L	Set A	Set B	N_L	Set A	Set B	N_L	Set A	Set B
10	(0,4,6)	(3,3,4)	15	(0,7,8)	(4,6,5)	20	(4,8,8)	(5,8,7)
11	(0,5,6)	(3,4,4)	16	(3,5,8)	(4,6,6)	21	(5,8,8)	(5,8,8)
12	(0,5,7)	(3,4,5)	17	(3,6,8)	(5,6,6)	22	(6,8,8)	(6,8,8)
13	(0,6,7)	(4,4,5)	18	(4,6,8)	(5,6,7)	23	(7,8,8)	(7,8,8)
14	(0,7,7)	(4,5,5)	19	(5,6,8)	(6,6,7)	24	(8,8,8)	(8,8,8)

Table 3: Optimal values of (N_y, N_b, N_r) for each value of N_{L}, the LUT size, under each point set. Set A consists of pixels extracted from an image in the sequence, while set B consists of uniformly distributed points in $\mathrm{YC_bC_r}$.

LUT shape for a set of M training points is given by

$$(N_Y^*, N_b^*, N_r^*) = \underset{(N_Y, N_b, N_r)}{\arg\min} \frac{1}{M} \sum_{i=1}^{M} D_{\mathrm{L}}(\mathbf{x}_i, \hat{\mathbf{x}}_i) \quad \text{subject to } N_Y + N_b + N_r = N_{\mathrm{L}}. \tag{7}$$

The restrictions placed on allowable values of N_Y, N_b, and N_r make feasible optimization of these values via exhaustive search.

The optimal values (N_Y^*, N_b^*, N_r^*) depend on the set of points used in (7). It is impractical to perform the optimization over every possible point in $\mathrm{YC_bC_r}$ since there are 2^{24} points. Instead, a set of M training points, $M \ll 2^{24}$, is used for optimization. The optimization procedure was performed for two different training sets. Set A consists of flesh and background points sampled from the first frame of *silent*, a standard MPEG-4 test sequence. Set B consists of 10 000 points uniformly distributed throughout $\mathrm{YC_bC_r}$. Table 3 shows the optimal LUT shapes for each point set for $10 \leq N_L \leq 24$ and indicates that the optimal shape depends on the test set. In general, the optimal LUT shape has more bits in the chrominance components and fewer bits in the luminance. This agrees with the above analysis that found that the luminance component contributes little to discrimination.

The LUT shapes that are optimal for set A are used in practice for segmentation, since they were extracted from an actual image in the sequence. The pixel color distribution encountered in practice, however, will generally differ slightly from the distribution of this set of points. It is essential that the performance of the optimized LUT does not significantly degrade in the presence of different pixel distributions. To measure the sensitivity to different distributions, a LUT for each value of N_{L}, $10 \leq N_{\mathrm{L}} \leq 24$, was optimized on set A and tested with set B. A large difference in performance would indicate that the optimal shape is sensitive to different distributions. Figure 2 shows the performance of the LUT optimal for set A tested with both set A and set B. The squares represent the distortion, defined in (7), of set A points; the circles represent the distortion of set B points. The plots indicate that the maximum difference in performance between the two point sets is 8.5%. When $N \geq 16$ the difference in distortion is less than 1%. These results suggest that classification performance will not degrade significantly for $N \geq 16$ when faced with a variety of pixel distributions.

Figure 3 shows the effect of different LUT sizes on segmentation. Results are shown for LUT sizes of 10 and 14 bits, as well as for the full 24-bit LUT. The figure illustrates that the segmentation improves when the LUT size increases from $N_{\mathrm{L}} = 10$ to $N_{\mathrm{L}} = 14$. Only minimal improvement occurs when N_{L} is further increased. The segmentation in Figure 3(e) uses the shape optimal for set B, and demonstrates the consequences of using inappropriate LUT shapes.

3. Video Coder Overview

Coding standards such as H.261 and H.263 are block-based.[21,22] Most implementations of these standards give equal importance to each block. While different blocks within the same picture may be coded with different modes, no one block is more important than another. This model is not appropriate for sign language. Blocks that correspond to hands and face are more important than blocks in the image background. Therefore, bandwidth should be allocated to achieve high quality for these perceptually important regions at the expense of background blocks.

One way to accomplish this goal is to give priority to a ROI. Figure 4 shows the process of coding an image with an ROI method. A classifier identifies the ROI in the image and the coder then allocates enough bits to the ROI to achieve a certain level of quality. The remainder of the image is coded with the remaining bits, possibly resulting

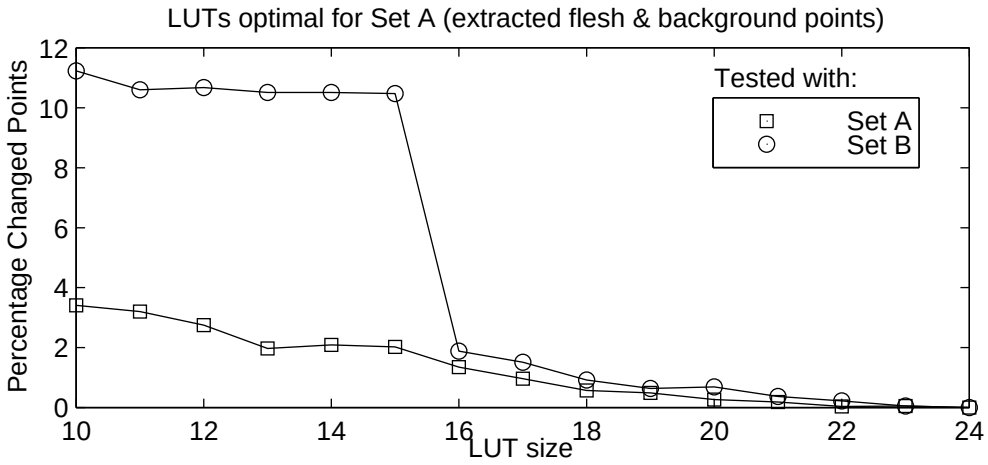

Figure 2: Distortion using LUT shapes optimal for set A, tested with points from set A (squares) and points from set B (circles). Set A consists of pixels extracted from an image in the sequence, while set B consists of uniformly distributed points in YC_bC_r.

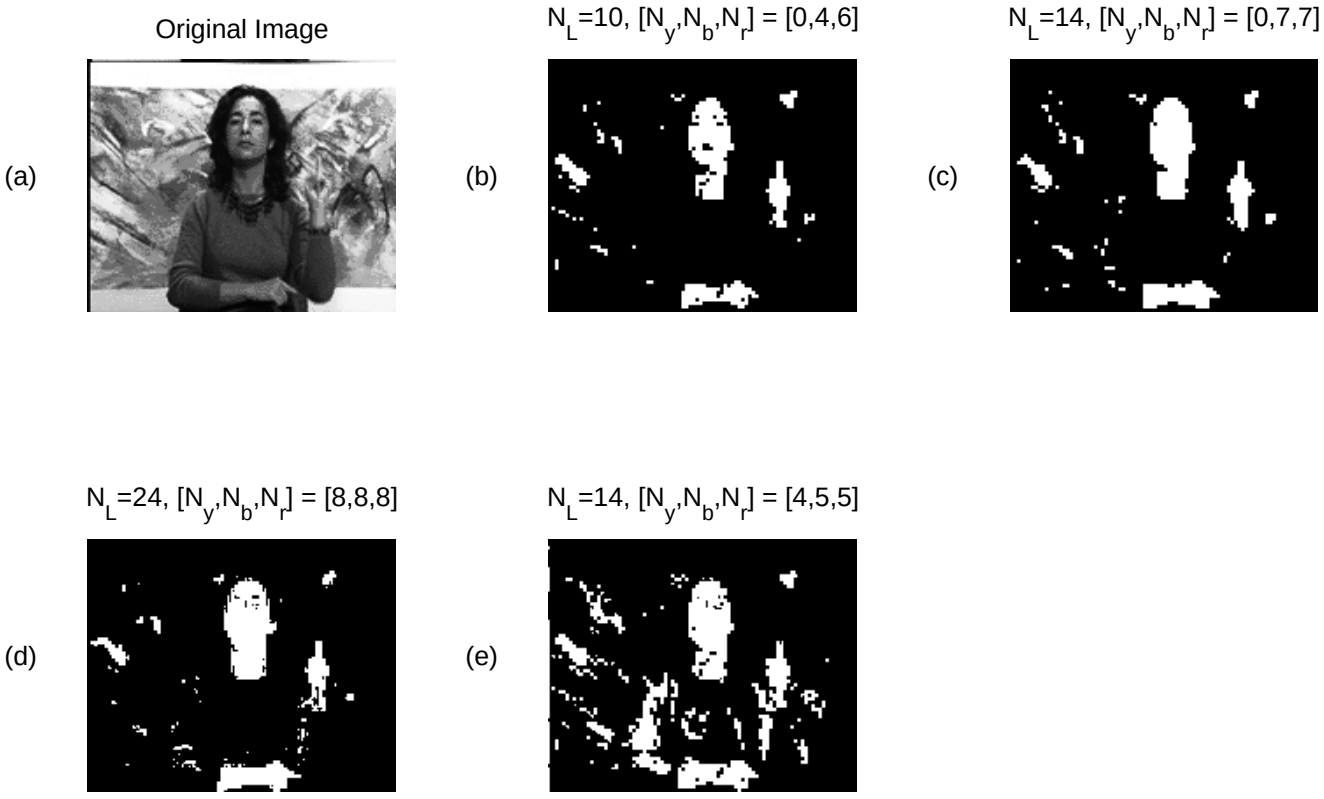

Figure 3: Images segmented with different $YC_bC_r \rightarrow a^*b^*$ LUT sizes. (a) Original image. Frames (b) and (c) use optimal LUTs for set A. Frame (d) uses the maximum size LUT. Frame (e) uses a LUT optimal for set B.

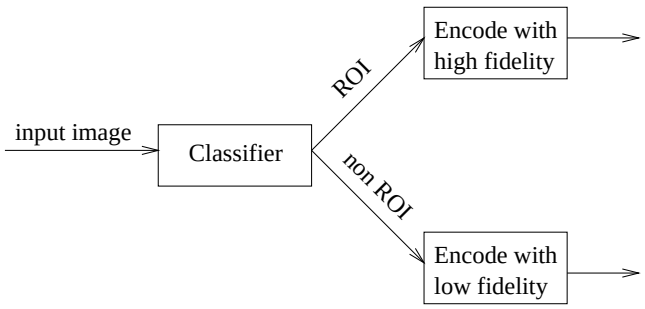

Figure 4: Image segmentation with ROI algorithm.

in lower quality for the pixels outside the ROI. A method for sign language should locate the face and hands in the image in real time and use the identified regions as the ROI.

Other authors have investigated video compression schemes that give priority to an ROI[1,2]. Many earlier proposals are inappropriate for sign language, however, since they use strictly the face as the ROI. The face segmentation methods in these proposals vary widely.[10,23,24] Moreover, many of these methods are incompatible with existing standards, and are computationally expensive; they cannot achieve high frame rates on current hardware. Exceptions are the methods in[25] and[26] that are compatible with H.261. In these methods, once an ROI is defined, quantizer values are adjusted to give more bits to the ROI. Neither method, however, took advantage of the ability to selectively encode macroblocks or attempted to control the frame rate.

The implemented video coder is a modified H.263 coder. Extensions to H.263 were necessary because the quantizer value Q cannot be specified at the macroblock level. The standard only allows for specification of ΔQ, which is limited to the range ± 2. This is unsuitable for the ROI algorithm, because ΔQ between ROI and non-ROI macroblocks can be much larger. The H.263 syntax was modified by inserting a macroblock map and an extra QUANT parameter. The macroblock map indicates whether each macroblock is ROI or non-ROI. The original QUANT parameter is used for the ROI region, and the additional QUANT is used for the non-ROI region.

The classification of macroblocks as ROI or non-ROI is a two-step process.[5] Individual pixels are classified as flesh color or non-flesh color using a Bayes classifier. The Bayes classifier requires estimates of the flesh and non-flesh color distributions. These distributions are modeled as Gaussian mixtures and are estimated from pixel colors in a pre-segmented training image. Morphological processing is then used to classify each macroblock.

Instead of Bayes classifying each image pixel in each frame, a subset of possible colors is classified during training and the results stored in a LUT, as described in Section 2.3. This greatly speeds the coding process. A full description of the color distribution estimation algorithm is given in.[5]

4. Results

Results of using the color-based classifier at the pixel level on a frame of *silent* are shown in Figure 3. This section presents results of coding video sequences with the developed algorithm.

An accurate ROI segmentation algorithm is necessary for an effective coder. If too many macroblocks are labeled ROI, then bandwidth will be wasted coding these blocks with high fidelity. If the classifier misses ROI macroblocks, then these important regions will be poorly coded. To measure the accuracy of the ROI classifier, the ROI decisions were compared to a manually classified sequence. Two methods for identifying the ROI were tested, and the results are shown in Table 4. The first method simply counts the number of flesh pixels in each macroblock, labeling it as ROI if the count surpasses a threshold. The second method uses morphological post-processing to achieve a more accurate ROI.[5]

The ROI algorithm improved the ROI quantizer value over the uniform algorithm while maintaining frame rate. Table 5 presents the results of coding the *silent* sequence using the coder at both 16 kbps and 32 kbps. The measured performance gains are confirmed through subjective evaluations. Figure 5 shows one decoded frame resulting from

ROI Identification	False Alarm	Miss
pixel count	2.3%	39.2%
post process	13.1%	10.5%

Table 4: Accuracy of automatic ROI identification algorithms versus a hand segmented sequence.

Algorithm	Target Bit Rate	PSNR Y	Mean Quant.	Mean Fr. Rate
Uniform	32 000	31.4	9.9	9.6
ROI	32 000	34.2	6.2	9.8
Uniform	16 000	28.2	17.4	9.4
ROI	16 000	29.5	13.4	9.1

Table 5: Data from modified H.263 coding of *silent*. The PSNR was calculated only for the ROI in each frame.

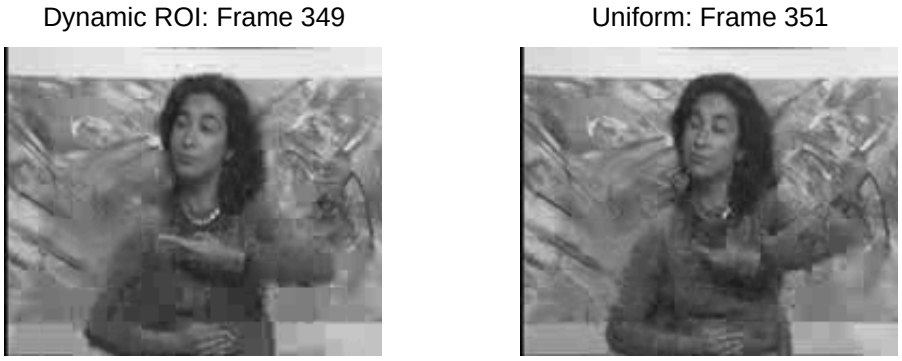

Figure 5: Comparison of coded frames of *silent* for the uniform and dynamic ROI algorithm, at 32 kpbs. The difference in frame number is due to the slight difference in frame rate.

encoding *silent* at 32 kbps. The face and fingers are noticeably clearer with the dynamic algorithm. The improved clarity is even more apparent when viewing the decoded sequences than when viewing a still frame.

Figure 6 shows the results of the block processing for several frames of *silent*. The left column shows the output of the Bayes classifier. Pixels identified as flesh are shown in white. While virtually all flesh areas are correctly identified, several background pixels are incorrectly identified as flesh. The center column shows the ROI macroblocks after the first part of the morphological processing. The false background areas are eliminated, along with portions of the head and fingers. The remaining morphological rules yield the ROI in the right column. The ROI now completely encompasses the face and hands, while still rejecting background areas.

5. Conclusions

The results of still-frame classification shown in Figure 3 demonstrate the effectiveness of using color to identify flesh-colored regions in an image. The same figures also confirm that the LUT configuration is a critical parameter. Inappropriate values for this parameter lead to degraded classifier performance. This parameter must be determined for each new combination of flesh and background colors.

The results also show that the presented coding algorithm addresses the requirements of sign language video. The face and hand location portion of the method is fast and accurate. The morphological macroblock processing identifies appropriate macroblocks for inclusion in the ROI. By allocating most of the available bandwidth to the ROI, the quality of the ROI is better than in the uniform algorithm.

Future work will improve the performance of the current algorithm in several ways. Tracking of flesh regions will

Figure 6: Results of the segmentation algorithm on five frames of *silent*. Left column: Identified flesh pixels shown in white. Center: Initial ROI macroblocks. Right: Final ROI macroblocks identified by post-processing.

result in fewer false ROI macroblocks as well as an improved motion compensation algorithm. Providing the decoder with ROI information will allow background restoration, eliminating the need to transmit background macroblocks.

6. REFERENCES

1. E. François, J.-F. Vial, and B. Chupeau, "Coding algorithm with region-based motion compensation," *IEEE Trans. Circ. Sys. Video Tech.* **7**, pp. 97–108, Feb. 1997.
2. H. Katata, N. Ito, and H. Kusao, "Temporal-scalable coding based on image content," *IEEE Trans. Circ. Sys. Video Tech.* **7**, pp. 52–59, Feb. 1997.
3. M. Lee, W. Chen, C. B. Lin, *et al.*, "A layered video object coding system using sprite and affine motion model," *IEEE Trans. Circ. Sys. Video Tech.* **7**, pp. 130–145, Feb. 1997.
4. R. Schumeyer, E. Heredia, and K. Barner, "Region of interest priority coding for sign language videoconferencing," in *IEEE First Workshop on Multimedia Signal Processing*, pp. 531–536, (Princeton), 1997.
5. R. Schumeyer and K. Barner, "Color-based content coding with applications to sign language video communications." Submitted to *IEEE Transactions on Circuits and Systems for Video Technology.*
6. R. Brunelli and T. Poggio, "Face recognition: Features versus templates," *IEEE Trans. Patt. Anal. Mach. Intell.* **15**, pp. 1042–1052, Oct. 1993.
7. R. Chellappa, C. L. Wilson, and S. Sirohey, "Human and machine recognition of faces: A survey," *Proc. IEEE* **83**, pp. 705–740, May 1995.
8. I. Craw, H. Ellis, and J. Lishman, "Automatic extraction of facial features," *Pattern Recognition Letters* **5**, pp. 183–187, Feb. 1987.
9. L. C. De Silva, K. Aizawa, and M. Hatori, "Detection and tracking of facial features," *Proc. SPIE* **2501**, pp. 1161–1172, 1995.
10. G. Bedini, L. Favalli, A. Marazzi, A. Mecocci, and C. Zanardi, "An integrated approach for high-compression of videoconference sequences," in *Proc. IEEE Intl. Conf. Comm. 95*, vol. 1, pp. 563–567, 1995.
11. R. R. Rao and R. M. Mersereau, "On merging hidden Markov models with deformable templates," in *Proc. ICIP 95*, vol. 3, pp. 556–559, 1995.
12. R. Talluri, K. Oehler, T. Bannon, *et al.*, "A robust, scalable, object-based video compression technique for very low bit-rate coding," *IEEE Trans. Circ. Sys. Video Tech.* **7**, pp. 221–233, Feb. 1997.
13. P. Duchnowski, M. Hunke, D. Büsching, U. Meier, and A. Waibel, "Toward movement-invariant automatic lip-reading and speech recognition," in *Proc. ICASSP 95*, vol. 1, pp. 109–112, (Detroit), 1995.
14. K. Sobottka and I. Pitas, "Extraction of facial regions and features using color and shape information," in *Proc. ICIP 96*, vol. 3, pp. 483–486, 1996.
15. C. H. Lee, J. S. Kim, and K. H. Park, "Automatic human face location in a complex background using motion and color information," *Patt. Rec.* **29**(11), pp. 1877–1889, 1996.
16. Y. Dai and Y. Nakano, "Face-texture model based on SGLD and its application in face detection in a color scene," *Patt. Rec.* **29**(6), pp. 1007–1017, 1996.
17. J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics: Principles and Practice*, Addison-Wesley, second ed., 1990.
18. A. K. Jain, *Fundamentals of Digital Image Processing*, Prentice-Hall, 1989.
19. K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, San Diego, second ed., 1990.
20. R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, John Wiley, 1973.

21. ITU-T, "ITU-T recommendation H.261: Video codec for audiovisual services at $p \times 64$ kbits," Mar. 1993.

22. ITU-T, "ITU-T recommendation H.263: Video coding for low bit rate communication," Mar. 1996.

23. A. Eleftheriadis and A. Jacquin, "Automatic face location detection and tracking for model-assisted coding of video teleconferencing sequences at low bit rates," *Signal Processing:Image Communication* **7**(3), pp. 231–248, 1995.

24. J. Luo, C. W. Chen, and K. J. Parker, "Face location in wavelet-based video compression for high perceptual quality videoconferencing," in *Proc. ICIP 95*, vol. 2, pp. 583–586, 1995.

25. E. Badiqué, "Knowledge-based facial area recognition and improved coding in a CCITT-compatible low-bitrate video-codec," in *Picture Coding Symposium*, 1990.

26. A. Eleftheriadis and A. Jacquin, "Low bit rate model-assisted H.261-compatible coding of video," in *Proc. ICIP 95*, vol. 2, pp. 418–421, 1995.