Efficient Image Segmentation Algorithm for Mobile Devices

Mark Smith
University of Central Arkansas
Conway, Arkansas 72035

Abstract

An efficient image segmentation algorithm utilized for mobile applications running on the iPhone's iOS platform is presented. Mobile devices such as the iPhone have limited CPU and memory resources, thus presenting a more challenging task when implementing complex algorithms such as image segmentation. The image segmentation utilized in this work splits the image into real-world objects that are numbered for the user to either select for further processing. First, a color quantization algorithm is applied to the entire image thus simplifying the image to only 16 available colors. Next, a fast texture measurement utilizing the co-occurrence matrix is applied to entire image using a pre-selected neighborhood of interest. Multiple regions are then automatically merged based on a color comparison measurement extracted at each object's boundary. The resulting regions are then displayed to the user for further analysis or selection. The primary usage of this algorithm is within other mobile apps that require the segmentation of images into realistic objects. Examples of these apps would be those that read bar codes, QVC codes, or OCR text regions. Results are shown for numerous standard image samples and compared with other image segmentation algorithms.

.

1. Introduction

The interest in mobile devices has exploded in recent years, especially the usage of the Apple's iPhone and iPad. These devices allow users to capture pictures and live video instantaneously while processing these images in real-time by an App. The App described in this paper is used for segmenting the image into real-world objects that can be used for further processing. In other words, this App could provide a foundation for other Apps that require an image (or even a video) segmented into realistic objects for further analysis. The image segmentation algorithm described in this work very efficiently processes the image thus minimizing the limited CPU and memory requirements. A novel image segmentation algorithm implemented in this app consists of the following steps:

- Color Quantization to 16 colors
- Fast Texture measurement extracted from the Co-Occurrence Matrix
- Adjacent regions are merged based on a dominant color matching. Algorithm.
- The resulting segmentation provides a realistic set of objects

The following sections outline each of these steps found in the process discussed in this work as well as a results section comparing this algorithm with other comparable systems.

2. Color Quantization and Color Matching

There are tens of thousands of unique colors in a given image and perhaps millions of unique colors across several pictures of a video sequence. The quantization of all possible colors to only a few levels is an important simplification step, since the comparison so many different color possibilities prove difficult when identifying the optimal foundation color to be applied to a region. The image undergoes a standard k-means clustering algorithm [9,17] and 16 quantized colors are extracted from this initial object. The motivation behind using 16 colors is because it has been found that most realistic regions can be represented by this many discrete labels - thus shading, textured regions, etc can be modeled most accurately this way. Before clustering, the original RGB pixel colors are converted to the CIE- L*a*b* color space which has been shown to be perceptually uniform and therefore preserve more accurate distances than the RGB color space, thus providing superior results [14]. The clustering results on the CIE-L*a*b* colors are then converted back to the RGB colors, the main feature used in this system.

The quantized colors in the regions are then compared with the actual colors in the other regions. The colors will be classified in one of two ways:

1. An existing color found in the largest possible region.
2. A new color not found in the region.

The symbol *pcn* will be used to represent the actual color in an additional region whereas *pcp* represents the corresponding matching color in the largest region. A new color is identified in the additional regions by (1) as

$$\| \mu\text{-}pcn\| > \max\|\mu\text{-}pci\| + \alpha\sigma \quad (1)$$

where μ is the mean of the cluster that *pcp* belongs to, pci is the *ith* color belonging to this cluster with $i = 1, 2, \ldots N$, and N is the total number of colors grouped with the cluster. σ is the standard deviation of the distances computed between μ and the colors in its cluster and is given by

$$\sigma = \sqrt{\frac{\sum_{i}^{N} \|\mu - pci\|}{(N-1)}} \quad (2)$$

and α is a scaling factor. We have found that α equal to 2 works well for the application considered in this work. This color-matching step is illustrated in Fig. 1.

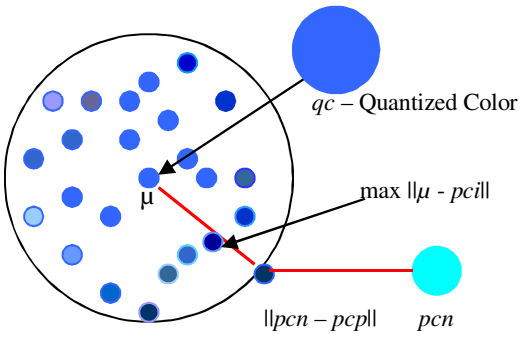

Figure 1. Color Matching

In the example shown in Fig. 1, *pcn* is classified as a new color.

3. Co-Occurrence Texture Measurement

The Gray-Level Co-occurrence Matrix (GLCM) is one of the most popular statistical texture measurements [15,19] and has been used as the primary component in a wide range of image segmentation applications [18,20]. The GLCM is a second-order statistical measurement; second-order statistics take into account the relationship between groups of two (usually neighboring) pixels in the original image. In contrast, first-order statistics, (e.g., mean and variance), do not consider any neighborhood associations. The process by which the GLCM is computed is outlined as follows

1. The GLCM computation utilizes the relation between two pixels at a time; one is called the reference and the other the neighbor pixel.
2. A displacement vector d, as specified as

   $d = < d_x, d_y >$ (d_x - distance in horizontal direction)

   (d_y - distance in vertical direction)

   is selected and determines the relationship between the pixels in the image. Utilizing only neighboring pixels (d = 1) is the most commonly used distance measurement and is also the one utilized in this system.
3. There are 8 possible relationships (i.e., displacement vectors) that can be formed between neighboring pixels (directions between neighboring pixels are shown in parenthesis –the first component refers to the horizontal displacement, whereas the second parameter refers to the vertical displacement):

   < 1,0> (0)
   < 0,1> (90)
   <-1,0> (180)
   < 0,-1> (270)
   < 1, 1> (45)
   <1,-1> (315)
   <-1, 1> (135)
   <-1,-1> (225)
4. A displacement vectors d is chosen for each co-occurrence matrix calculation [4,11]. All occurrences of gray levels i and j of two pixels separated by displacement vector d are accumulated. For instance, if $i = 0$, and $j = 0$, and the displacement vector is <1,0>, the calculation is performed by accumulating the frequency on the selected image region that a pixel with gray level 0 (neighbor pixel) falls to the right of another pixel with gray level 0 (reference pixel). The GLCM is a very compact and optimal measurement. An example illustrating a complete GLCM calculation, consider the following 4x4 image with pixel values shown in Fig. 2:

0	0	1	1
0	0	1	1
0	2	2	2
2	2	3	3

Fig. 2 Example 4x4 Image Gray-Levels

For example, if the East displacement vector is chosen (i.e., <1,0>), each image pixel is selected in turn as a reference pixel. The pixel immediately to its right is then chosen as the neighbor pixel. The occurrences of these two pixels together are then accumulated. In this example, 0-0 occurs twice, 1-1 occurs twice and so on. The entire co-occurrence matrix for the image in Fig. 2 and the <1,0> displacement vector is shown in Fig. 3.

	0	1	2	3
0	2	2	1	0
1	0	2	0	0
2	0	0	3	1
3	0	0	0	1

Fig. 3 GLCM Computed for Fig. 2

Both the rows/columns pertain to a discrete gray-level 0,1,2, or 3. Note that the co-occurrence matrix is square and its dimensions are always determined by the number of gray-levels (i.e., for this system number of quantized colors) of the image [6,13]. The GLCM dimensions are CxC where C is the largest gray-scale value, or number of quantized colors.

4. Image Segmentation Algorithm

This system uses the mean of the GLCM as a key feature in its image segmentation algorithm (see section 4.) and the mean of the GLCM and its magnitude is given below as [12]:

$$\mu_i = \sum_{i=1}^{C}\sum_{j=1}^{C} i(P_{ij}) \quad (3)$$

$$\mu_j = \sum_{i=1}^{C}\sum_{j=1}^{C} j(P_{ij}) \quad (4)$$

$$\| \mu \| = \sqrt{\mu^2{}_i + \mu_j{}^2} \quad (5)$$

Where μi is the horizontal mean, i is a given row value, Pij is an element of the GLCM, μj is the vertical mean, j is a given column value, ||μ|| is the mean's magnitude, and N is the size of the sliding window used in computing the GLCM mean. The two values, μi and μj are equal because of the GLCM's symmetry [7]. Plotting the magnitude of the GLCM's mean as a 2-d image is shown below in Figure 5:

Fig. 4. GLCM Mean Feature

The mean textured image is very smooth and almost all micro-textures have been removed [8]. The individual region interiors possess consistent gray-scales throughout this image; therefore the region boundaries can be identified from a basic edge detection filter. An NxN filter computing the variance as is selected as the edge detection filter and is convolved with the mean textured image resulting in an edge intensity image. In equation (4) above, μ is the average grayscale within the NxN region and gij is the gray-scale at the ith row, jth column of the GLCM mean. A plot of the edge image generated as a result of applying the above filter is shown below in Figure 5.

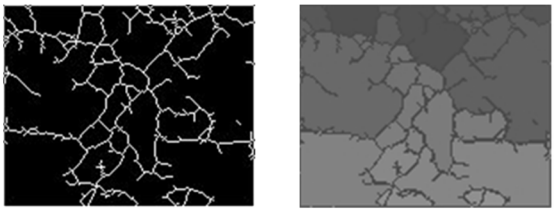

Fig. 5. Image Segmentation Results

The highlighted regions provide an initial set of objects.

5. Region Merging Algorithm

The objects created from the image segmentation of section 3 are occasionally over-segmented thus requiring an additional step to merge similar objects together [10,16]. A step that merges smaller regions with the larger, adjacent region is needed to provide optimal object segmentation. The region merging algorithm introduced in this section demonstrates that small color samples extracted near the boundaries of adjacent regions provide an excellent criteria for merging the areas. The algorithm utilized in this system relies on the dominant (quantized) colors when comparing adjacent regions. Therefore, the adjacent regions are merged based on how similar their colors are to the largest region. The example shown below is for a standard image. The algorithm is summarized as:

1. The regions created by the image segmentation are extracted. The regions (and their corresponding labels) as well as their contours overlaid onto the original color frame are shown in Figure 3.

2. All neighboring segments for each region are determined and only those neighboring segments that are larger are considered as merging candidates. The main concept is that smaller regions are only merged with larger, bordering regions. For example, region 12 has larger

neighboring segments 5,11,16 and 17, whereas region 17 has larger adjacent segment region *5*.

3. Each region's quantized colors are then compared with the quantized colors of each of its larger, neighboring segments. The smaller region will be merged with the larger one if their quantized colors are sufficiently close [5]. The steps utilized in this process are outlined as follows:
   a. A windowed area running the length of the adjacent boundary between neighboring objects is selected for each region. Each area provides a representative sample of the quantized colors for the object. Colors selected at their adjacent boundary provide the best measurement on whether the objects should be merged, thus minimizing the effects from outlying colors. The sampled regions usually have a maximum width of 5 pixels and are parallel to the entire length of the boundary. Additional points are selected when the sampled regions consist of 25 pixels or less. Examples of these sampled regions are shown in Figure 6 for selected neighboring objects.

Fig. 6. Selected Regions

   b. Each quantized color (i.e., discrete label) and its corresponding concentration (measured in percentage) are extracted from each sampled area within each region. Only those quantized colors with a concentration greater than 5% are considered.
   c. If the majority of the quantized colors of the smaller region match those of the larger region, the larger region is then selected as a candidate for merging with the smaller one.
4. Step 3 is repeated for all larger neighboring objects and all candidates for merging with the smaller objects are maintained [2].
5. The candidate which best matches the smaller object's quantized color concentration is then selected as the best matching region for merging. The smaller region is then marked for merging with the larger region – but the actual object merging is not done at this time.
6. Steps 1 – 5 are repeated for all remaining objects.
7. All smaller objects previously marked for merging are then merged with their best matching neighboring objects.

The results of this algorithm as applied to the original segmentation, Figure 3, is shown in Figure 5.

Fig. 7 Region Merging Results

6. Results

The image segmentation algorithm described in this work was implemented as an App on Apple's iPad iOS 7.1 platform using XCode and Objective-C. Testing the App was performed using a series of standard test images added to the iPad's camera roll [9]. The test pictures consisted of popular images extracted from three different categories – *Happy Granny, Foreman*, and *Tennis*. Examples of these test images are shown below:

Fig. 8. Test Images

The author's algorithm was compared with 2 other popular image segmentation algorithm commonly referenced in the literature. The first of these was the JSEG [1] algorithm developed by researchers at UCSB while the second algorithm is implemented as part of the OpenCV [3] library, a popular image processing library implemented by Intel. The algorithms utilize both color and texture when segmenting images. The algorithms were compared based on their speed in milliseconds required to process each of the test images on the iPad device. A table showing the results of these comparisons is shown below in Table I:

Table 1

Algorithm	*Happy Granny*	*Foreman*	*Tennis*
Author's	875	1478	583
JSEG	923	1367	1033
OpenCV	2154	3382	1932

As observed from the table, the results of the author's algorithm appear to be very promising. The algorithm described in this work could provide the substrate layer needed for many apps implemented on the iOS platform requiring a captured image segmented into realistic objects. The applications for this type of algorithm is numerous, especially when segmentation of specific regions such as barcodes or printed text is required.

Future work for this system include enhancing various apps (e.g., barcode, text) with the image segmentation algorithm described in this work thus providing them with the efficient object segmentation capabilities often only currently found in high-end desk-top applications
:

7. References

[1] Y. Deng and B. S. Manjunath, "Unsupervised segmentation of color-texture regions in images and video," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 22, no. 6, pp. 939-954, 2001.

[2] Air Pressure: Why IT Must Sort Out App Mobilization Challenges". InformationWeek. 5 December 2009.

[3] E. D. Gelasca, E. Salvador and T. Ebrahimi, "Intuitive strategy for parameter setting in video segmentation," Proc. IEEE Workshop on Video Analysis, pp.221-225, 2000.

[4] MPEG-4 , "Testing and evaluation procedures document", ISO/TEC JTC1/SC29/WG11, N999, (July 1995).

[5] R. Mech and M. Wollborn, "A noise robust method for segmentation of moving objects in video sequences," *ICASSP '97 Proceedings*, pp. 2657 – 2660, 1997.

[6] T. Aach, A Kaup, and R. Mester, "Statistical model-based change detection in moving video," *IEEE Trans. on Signal Processing*, vol. 31, no 2, pp. 165-180, March 1993.

[7] L. Chiariglione-Convenor, technical specification *MPEG-1 ISO/IEC JTC1/SC29/WG11 NMPEG 96*, pp. 34-82, June, 1996.

[8] MPEG-7, ISO/IEC JTC1/SC29/WG211, N2207, Context and objectives, (March 1998).

[9] P. Deitel ,*iPhone Programming*, Prentice Hall, pp. 190-194, 2009.

[10] C. Zhan, X. Duan, S. Xu., Z. Song, M. Luo, "An Improved Moving Object Detection Algorithm Based on Frame Difference and Edge Detection," 4th International Conference on Image and Graphics (ICIG), 2007.

[11] R. Cucchiara, C. Grana, M. Piccardi, Member and A. Prati, "Detecting Moving Objects, Ghosts, and Shadows in Video Streams," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 25, no. 10, pp. 1337-1342, October, 2003.

[12] F. Rothganger, S. Lazebnik, C. Schmid and J. Ponce, "Segmenting, Modeling, and Matching Video Clips Containing Multiple Moving Objects," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29, no.3, pp. 477-491, March 2007.

[13] Neil Day, Jose M. Martinez, "Introduction to MPEG-7", ISO/IEC/SC29/WG11 N4325, July, 2001.

[14] M. Ghanbari, Video Coding an Introduction to standard codecs, Institution of Electrical Engineers (IEE), 1999, pp. 87- 116.

[15] L. Davis, "An Empirical Evaluation of Generalized Cooccurrence Matrices," IEEE Trans. Pattern Analysis and Machine Intelligence, vol 2, pp. 214-221, 1981.

[16] R. Gonzalez, Digital Image Processing, Prentice Hall, 2nd edition, pp. 326-327, 2002

[17] K. Castelman,Digital Image Processing, Prentice Hall, pp. 452-454, 1996.

[18] L. S. Davis and S. Johns, "Texture analysis using generalized co-occurrence matrices, " *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol 3, pp. 251-259, 1979.

[19] L. S. Davis, "An Empirical Evaluation of Generalized Cooccurrence Matrices," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 2, pp. 214-221, 1981.

[20] J. Haddon and J. Boyce, "Image segmentation by unifying region and boundary information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, October 1990.