

Optimal Luminance-Chrominance Downsampling through Fast K-Means Quantization

Nicolaie Popescu-Bodorin¹, Luminita State²,

¹*Dept. of Mathematics and Computer Science, Spiru Haret University, Bucharest, Romania, bodorin [a] ieee . org*

²*Dept. of Mathematics and Computer Science, University of Pitesti, Pitesti, Romania, radus [a] sunu . rnc . ro*

Abstract

This paper presents an algorithm within the k-means paradigm optimized for fast image processing and its applications to the downsampling of the luminance-chrominance channels of continuous-tone still images and to the iris segmentation.

Entropy reduction of any given image is achieved here by applying three main processing operations on each of the luminance / chrominance channels: a preliminary scalar quantization with 256 equally spaced reconstruction levels; a k-means computation of a new set of reconstruction levels containing fewer, non-uniformly spaced but much more significant values; the final quantization step using this second set of reconstruction values. No explicit median filter is applied.

This approach enables us to find the appropriate reconstruction values according to the image's chromatic features (gradient and edges) without explicitly computing them, while preserving local visual significance and the quality of the processed picture. Our results shows that severe downsampling rate (for example 1:1/16:1/16:1 downsampling corresponding to 256:16:16 YCbCr reconstruction levels) can be achieved with no significant loss in the visual quality of the processed image by applying the proposed Fast-K-Means Quantization algorithm. Finding 16 reconstruction values in each of the CbCr channels through Fast K-Means algorithm is achieved here with a computational cost comparable with that of the RGB to YCbCr conversion.

Keywords

iris segmentation, k-means, run-length filtering, fast k-means quantization.

1. Problem Statement

To avoid any confusion we will remind that the terms downsampling / downsizing / subsampling (and sometime resampling) are used in image processing to identify *a procedure by which the spatial resolution of an image is reduced* [1]. At first sight, there is no connection between downsampling and quantization defined as being the process in which *the amplitude of a signal is compared to a set of decision levels* [2]. But in fact, downsampling decreases the storage of an image creating (and ignoring) gaps in the original image, gaps that will be further reconstructed by some median filter. Consequently, downsampling/upsampling procedures retain in the image storage other values than those originally being stored by initial quantization of the image. This fact that implicitly links together the two discussed concepts enables us to formulate the following problem: *for a given image is it possible to identify a set of reconstruction values and a set of neighborhoods, both of the same length, ensuring that the mean of each neighborhood is the corresponding reconstruction value?* Theoretically, if it is possible, it means that we have found both a quantization and a median filter whose results are perfectly matching each other. More than that, from a practical point of view, we can expect that some morphological properties of the original image will become accessible (visible, at least) in the quantized image.

Further in this paper we will prove that k-means algorithm gives affirmative answer to the above question. Also, as a practical outcome, we will present two applications of Fast K-Means Quantization Algorithm to Luminance-Chrominance Downsampling and to Iris Segmentation.

2. Preliminaries

The classical approach to downsampling [11] is to consider the minimum coded units as being blocks, or much more generally neighborhoods, sometimes mutually exclusive, sometimes not, but small enough to minimize the visual effects of some median filter applied to them. In this way, the initial image is viewed as being covered by a uniformly spaced grid defining the blocks or by a uniformly shaped collection of neighborhoods. A lucky guess of such a cover ensures that chromatic variation across these neighborhoods is smaller than a desired threshold. Otherwise, an important (noise-like) local chromatic variation corresponding to an essential detail may produce a strong unwanted visual effect.

Also, downsampling can be done adaptively to amplitude of the high frequency components [3] and also by varying the threshold and the type of neighborhoods according to local chromatic variation. By doing so, we negotiate between making a more accurate numerical representation of any particular processed image and preserving enough regularity in the neighborhoods shape to ensure computational efficiency. Even so, while remaining an explicit median filtering on imposed uniformly shaped neighborhoods, adaptive downsampling doesn't solve the following issues:

- The encoding of one fixed chromatic value depends on the local chromatic medians computed across those neighborhoods containing that value instead of depending on the value itself. In this way, we are reducing the entropy in each of the minimum coded units but we are introducing in the encoded image an entropy that wasn't really there at the beginning (an intrinsic noise of the median filter that has been applied).
- By varying the threshold or the shape, dimensions and extent of the neighborhoods, we generate an encoding rule that is depending on the pixel position. Consequently, coding the encoding rule in a compressed standardized format becomes a challenge in itself.
- No matter how strong the visual effects are, median filtering on imposed uniformly shaped neighborhoods will always produce new edges placed on the each border of the filtered neighborhood. Surprisingly, we hope to preserve significant edges in the original image by introducing some new edges in the filtered image. This is also another noise, intrinsic to median filters, whose visibility is increasing with the regularity of the neighborhoods collection covering the image and also with the extent of these neighborhoods.

Of course, when speed is the main issue in processing, we don't bother about the fact that we will eventually see, in the downsampled image, things that weren't really there in the original image [9,10].

On the other hand, the simplicity level of the classical downsampling methods based on median filtering is difficult to achieve in other downsampling approaches. Consequently, a good negotiation between precision and simplicity becomes our target.

3. Principles of k-means optimal downsampling

This paper is an outcome of a few experimental studies that have been done by us following some new proposed principles of k-means optimal downsampling, principles suggested step by step by our practice and set out as follows:

- Any image is a self-described package of information, each chromatic layer being an array of pixels with random chromatic values for which the location is not important (when we are downsampling an image, in fact we are truncating the histogram to get more redundancy in order to reduce the entropy);
- During the downsampling process it is mandatory to maintain a functional correspondence between the chromatic values within original image and those within processed image. Doing otherwise is against the first principle from above and also, from a practical point of view, we implicitly add in the downsampled image a description of some facts that never existed in the original image.
- In the k-means context, the set of optimal reconstruction values (of a quantization at any given number of chromatic levels) and also the shape, dimensions and extent of the elements within the neighborhoods set covering the image are all intrinsic properties of the given image and consequently need not be imposed during the computation (we will let the image to talk about itself).

- Any robust efficient implementation of k-means algorithm designed to process at least a subclass of 8-bit/channel images must support uint8¹ acceleration over that subclass, or in other words, the normalized double representation of the processed image must be a stability point of computational mechanism tolerating at least round-off errors perturbations, meaning that for a given image and for a given choice of initial set of centroids, if we do substitute the uint8 representation for the double normalized one, the algorithm must converges nearly to the same solution (to a nearly optimal solution).

These principles had enabled us to formulate the Fast K-Means Quantization algorithm, an efficient iterative procedure in which:

- The classified clusters are the optimal choice of neighborhoods set computed without imposing restrictions on their shape, dimension and extent;
- The centroids of the clusters are the *optimal* choice of reconstruction levels that *is minimizing the within-cluster sums of point-to-centroid distances*.

4. Generic K-Means Algorithm

The problem formulation had suggested from the beginning that k-means algorithm is the first candidate for answering our question formulated in the first section of this paper. A succinct presentation of k-means algorithm is the following:

Generic K-Means Procedure:

```

INPUT: dataset to clusterize, desired number of clusters,
          eventually other custom data (such initial choice of centroids);
While the termination condition isn't satisfied:
  For each element in the dataset:
    Find the closest centroid (ambiguity must be treated also);
    Mark the appartenance of the element to the corresponding cluster;
  EndFor;
  If none of the elements in the dataset changes its appartenance then
    Fulfil the termination condition;
  EndIf;
  For each of the computed clusters:
    Recompute cluster's centroid as being the mean of contained
    elements;
  EndFor;
EndWhile;
END.

```

The results obtained by applying this algorithm on luma-chroma representation of 24-bit images normalized in double precision were very good, almost indistinguishable from the original images in many cases. The big problem is that the generic algorithm converges very slowly due the fact that very fine tuning of the values of the centroids is

¹ Matlab naming convention for 8-bit unsigned integer data type

done in a large number of iterations through a large dataset. When we were working with normalized double-precision images of dimension 512x512, our first m-script implementation of the generic k-means algorithm has spent over 20 seconds to find 16 to 32 clusters in one chroma channel. Therefore, the next step was to move the computations in uint8 domain, as much as possible.

5. Fast K-Means Image Quantization Algorithm

The main algorithm presented in this paper is an adaptation of the generic k-means algorithm, tuned for fast chromatic clustering in Matlab. It takes all computational advantage of working with uint8 representation of 24-bit images:

Collapsing memory storage by avoiding chromatic values redundancy: as we can see in the generic k-means algorithm, there is no need to keep trace of the pixel position in the image. This enables us to exploit the big redundancy of chromatic values which will make the image's storage to collapse. In the k-means context, there are only two important data to retain from the each luma-chroma plane of the input image: what unique chromatic values it contains and how many times they appear. Consequently, the exact histogram of the given chromatic plane is the proper storage to work with (is the shortest container for the two required pieces of data). All the information we need from the original chromatic plane could be contained in a maximal 2x256 uint8 array, no matter how big the original image. Consequently, the initial storage is read only once prior to k-means iterations.

Collapsing memory storage by avoiding appartenance index redundancy: initially, the index memorizing the appartenance of each pixel to one specific cluster is an two dimensional array as wide as the original image is, but again, its values depends on the chromatic values of the corresponding pixels instead of depending on the pixels position in the image. Therefore, the initial appartenance index can also be efficiently stored in a maximal 2x256 uint8 array containing duplicate data of the first part of the exact histogram (maximum 256 unique values) and corresponding appartenance index of each chromatic value.

The algorithm also make the most of Matlab acceleration mechanisms such as vectorized [6] optimized loops and efficient handling of large spreaded array subsets (loops elimination) through the use of logical indexing [7]. At first sight, these optimizations seems to add unnecessary (or even unacceptable) particularization of the algorithm, but all of these mechanisms are replicable, even with better time performances, in other programming languages.

Taking into account that, latter in this paper, we will be interested in estimating the algorithm's performance, isolating the true computational kernel from input, output and collapsing operations is the proper way to formulate the algorithm:

```
function RCP = resample_kmeans_uint8(CP, NC):  
    [EH, OC] = exact_histogram(CP);  
    % deflates chroma plane in its exact histogram;  
    [C, CI] = kernel_kmeans_uint8(EH, OC, NC);  
    RCP = parse_output(CP, CI, EH, C);
```

```

% inflates EH and CI to reconstruct chroma plane;

function [C, CI] = kernel_kmeans_uint8(EH, OC, NC);
    CI = uint8(zeros(size(EH'))); C = uint8([]); CC = uint8([1:NC]);
    T = false;
    while ~T
        C = get_new_centroids(C, EH, OC, CI, NC, CC);
        [T, CI, CC] = recompute_clusters(EH, OC, C, CI, T);
    end;

```

where the meaning of each variable is as follows:

RCP – reconstructed chroma plane array;

CP – chroma plane array;

NC – desired number of clusters;

EH – array of values appearing in exact histogram of CP;

OC - array of occurrences accounted for each of the values of EH;

C – array of computed centroids;

CI – collapsed cluster index;

T – termination flag;

CC – index of changed clusters

The main improvement against the generic k-means algorithm is the fact that, no matter how big the initial image, the computational kernel here is working with only a maximal set of 1024 values for each luma-chroma plane: maximum 3×256 uint8 values for EH, CI and C (but usually NC is small and consequently C is also shorter than EH), and maximum 256 unsigned integer values (uint32 for very large images such as 65536×65536 images) for OC. Therefore, the computational complexity of the computational kernel *does not depend on the size of the processed image*.

Taking into account that, in the reconstructed chroma plane, each chromatic value is replaced by the centroid of the cluster containing it, the fast k-means algorithm formulated above requantize the original chroma plane using the centroids of computed clusters as reconstruction values, as is shown in the following figure:

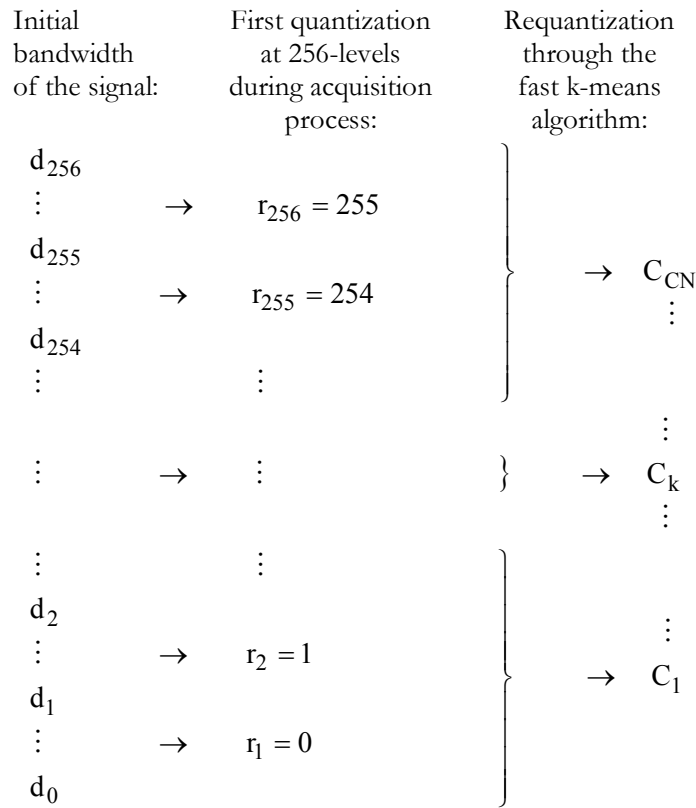


Figure1 Image requantization through fast k-means algorithm.

Being a quantization, k-means algorithm is implicitly an approximative reading operation on the original data, something that usually median filters aren't.

It is generally accepted [2] that in some ideal conditions (almost never satisfied by k-mean clustering mechanism) the optimal choice for the reconstruction values of a uniform scalar quantization is to place them in the middle of each decision interval. Our k-means requantization here certainly doesn't do that at all. The difference between the k-means computed centroids and those "ideally placed" is a measure of how ideal the real images aren't, and also for how un-natural are some ideal statistic criteria to human eye and to the image itself. A telling example can be seen in figure 2.

Another important aspect is that the Fast K-Means Image Quantization Algorithm tends to 'recognize' clusters that are also meaningful to human understanding of the picture. This is because the algorithm tends to ignore relatively constant chromatic areas in direct relation with their extent and with their position in their cluster: a relatively small area with relatively constant chromatic usually doesn't have an important contribution to the mean of the cluster containing it, especially when the values in that area are very different from the cluster's centroid.

All of the above observations and also the simplicity of the Fast K-Means Image Quantization Algorithm suggest that, at least in chromatic clustering, a good strategy is to assume that an image is a collection of pixels with no specific properties and to keep considering that until the moment when the image itself will tell us something else.

Taking other way means to overwrite the initially available information within the original image hoping to obtain better results from ideal hypothesis than from approximative reading.



Figure 2 A 4-means quantization example (original from CASIA V1 iris database).

On the other hand, the distance from ideal case to real image is not very big suggesting that the number of k-means iterations will be smaller enough when idealized reconstruction values will be used as initial choice for centroids. But when timing is critical, even a uniform partition from minimum to maximum of the histogram is a good choice for the initial values of the centroids. Our practice shown that few additional k-means iterations run incomparably faster than fine tuning in search of some decision intervals on which to ensure a constant probability density of the chromatic values [2].

6. Application of the Fast K-Means Image Quantization Algorithm to the Luminance-Chrominance Downsampling

The Fast K-Means Image Quantization Algorithm was initially designed for studying the entropy of 24-bit RGB images while we were searching for any kind of transform that could decrease image's entropy and could preserve visual quality of the processed picture also, especially for such kind of transform that could homogeneously spread the error (the difference between the original and the reconstructed image) over an irregular set of neighborhoods instead of accumulating it near a rectangular grid (part of the blocking effect in DCT²-coded images).

Visual results of the Fast K-Means Image Quantization Algorithm are usually very good, as it can be seen in the figure 3.

² Discrete Cosine Transform.

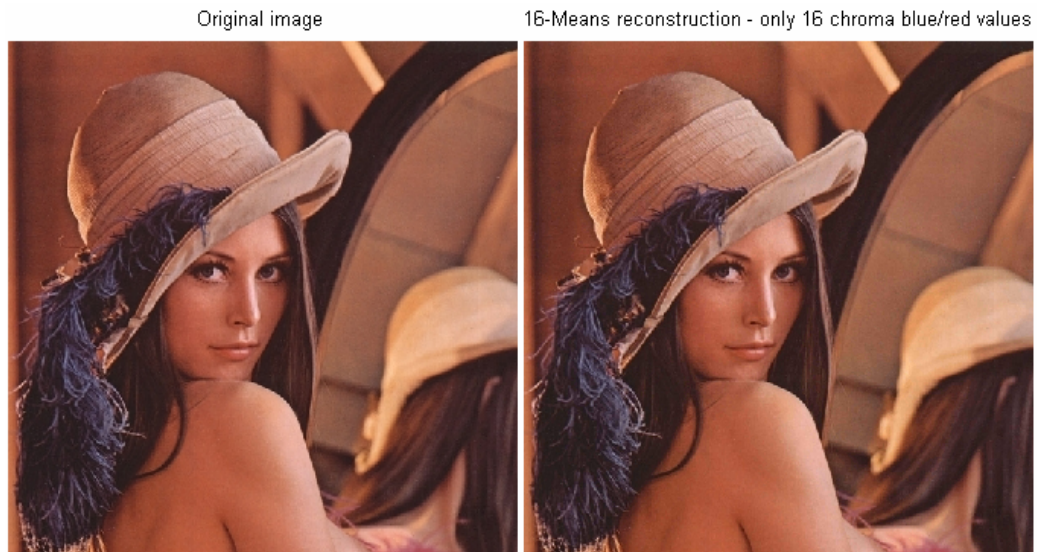


Figure 3 Fast 16-Means Chroma Quantization of Lena image (256 luma levels, 16 chroma blue/red levels).

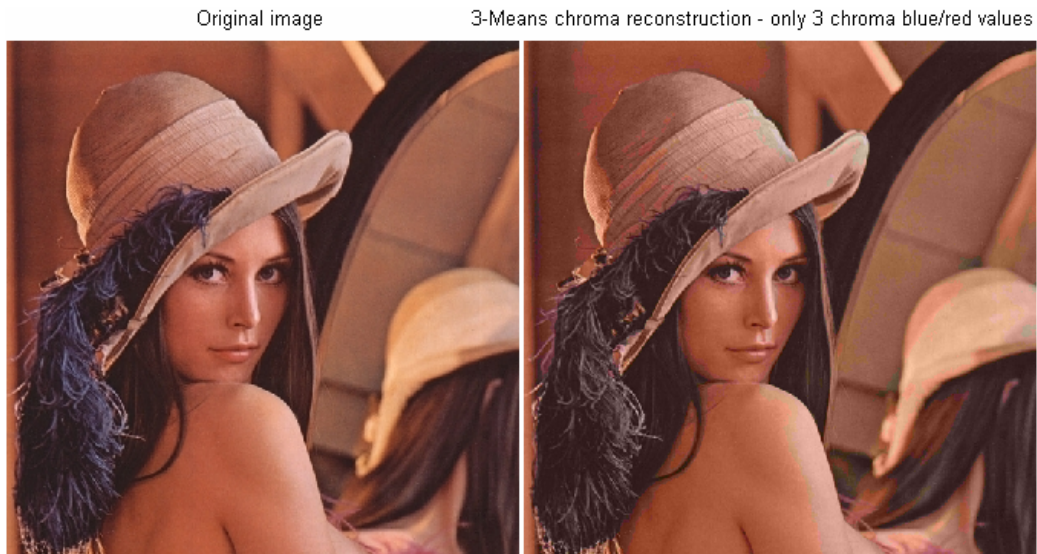


Figure 4 Fast 3-Means Chroma Quantization of Lena image (256 luma levels, 3 chroma blue/red levels).

Decreasing of the visual quality becomes evident for severe reduction of chroma levels number (figure 4) but, as expected, the visual quality is more sensitive to luma requantization (figure 5).



Figure 5 Fast 16-Means Luma-Chroma Quantization of Lena image (16 luma levels, 16 chroma blue/red levels).

All the benchmark tests that we have done so far includes many usually standard RGB-to-YCC and YCC-to-RGB conversions such as HDTV-709, PC-709, SDTV-601, PC-601, VIDEO-601, JPEG-601, PALTV, SECAM, Matlab implicit conversion, and other custom experimental conversions. A comprehensive m-script including the most of them will be available on the internet [8] as soon as possible, in uint8 version.

There is a sufficiently large class of images which are indistinguishable from their 256/16/16 luma-chroma reconstructions. For this kind of images, all the information in the chroma blue/red channels can be stored in one single channel (8 bit/pixel without chroma downsizing). Therefore, a first possible application is to use one 8-bit channel to store something else such an infrared image of the same subject, or the scaled response of the surroundings to a radar signal, or prediction values for the next frame, or any other data that can be fitted in the uint8 domain.

7. Application of the Fast K-Means Quantization Algorithm to the Iris Segmentation

The most important application of the Fast K-Means Image Quantization Algorithm that we have been done so far is the iris segmentation.

This section of the present paper uses CASIA V1-3 iris databases [12]. We express our gratitude to Chinese Academy of Sciences, Institute of Automation, for giving us permission to work with these databases. There are also few images from the old CASIA V1 database (in which the pupil isn't filtered) that we are working on.

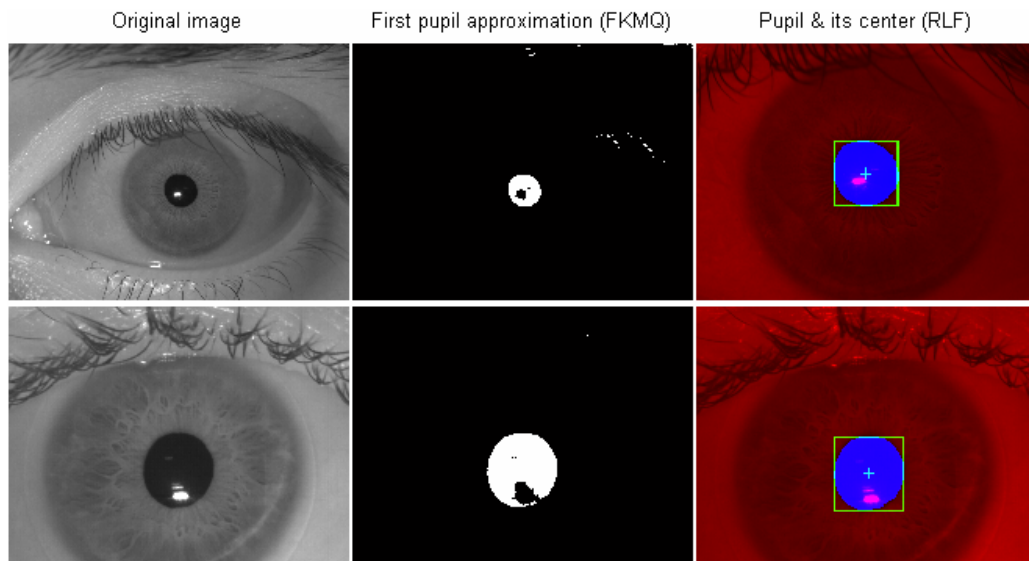


Figure 6 Fast 16-Means Iris Segmentation: finding the pupil and its center through Fast-K-Means Quantization and Run-Length Filtering (two images from old CASIA V1).

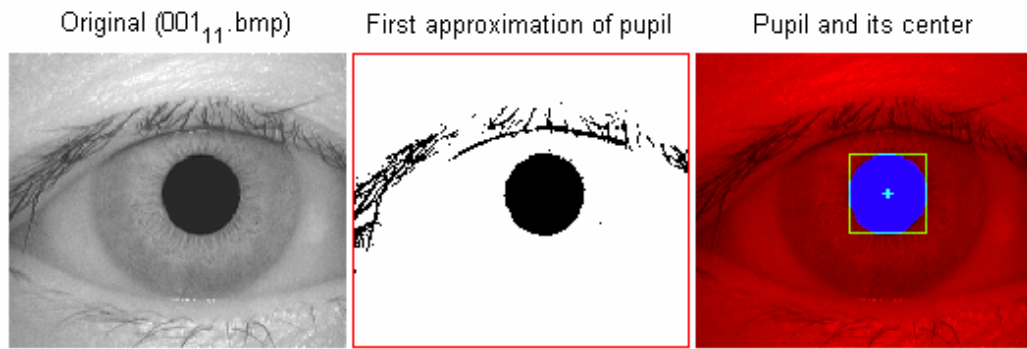


Figure 7 Fast 3-Means Iris Segmentation: finding the pupil and its center through Fast K-Means Quantization and Run-Length Filtering (image from CASIA V1).

We consider the generic process of iris recognition [4] as being divided into the following parts:

- 1) acquire the image;
- 2) crop the acquired image to significant area;
- 3) locate the pupil and its center;
- 4) do the iris segmentation;
- 5) detect and reject cases of visual exposure inconsistency;
- 6) compute the Key Features Array from extracted iris area;
- 7) compare the Key Features Array against those stored in some ID Keys Database;

This paper investigates especially localization of pupil and its center, and also the next two steps.

Initially the pupil localization has been done for some images from old CASIA V1 database (like those in the figure 6) and this has been proved to be simple enough to achieve using the proposed Fast K-means Quantization Algorithm which converged in a very few iterations to a very good approximation of the pupil area (the target signal) slightly perturbed only by some specular light and by some eyelashes points heaving chromatic values among the values of the pupil cluster. Here both specular light and eyelashes have the meaning of a noise. The signal to noise ratio being sufficiently good there was no problem to filter the unwanted noise. We have used Run-Length Encoding to filter the noise, deleting pixels around the pupil (erosion) and dilating the remaining cluster to fit its interior (or to close its interior when the specular light perturb the pupil's border, or both of the cases). Such a filter can be easily derived from any usual Run-Length Encoding procedure.

In the context of finding the pupil location through k-means algorithm, the present CASIA V1 database has been proven to be more challenging. This is because of the fact that the images are pre-processed, meaning that the pupil is filtered and the specular light on the pupil isn't present. There are two kind of data losses involved in this operation: the specular light can't be further used as an estimation of pupil location (whenever it exists, the specular light on the pupil is the easier area to locate in the entire image), and the second one - and the more important thing, the chromatic values in the pupil area becomes closer to those of the eyelashes. Therefore, the first approximation of the pupil computed through Fast K-Means Quantization Algorithm becomes noisier (figure 7) and all the segmentation procedure had to be recalibrated. All in all, the present V1 database has been proven to be more demanding when it came to test robustness of our segmentation algorithm. For all of these reasons from above, we have chosen the present CASIA V1 database to work with in this paper.

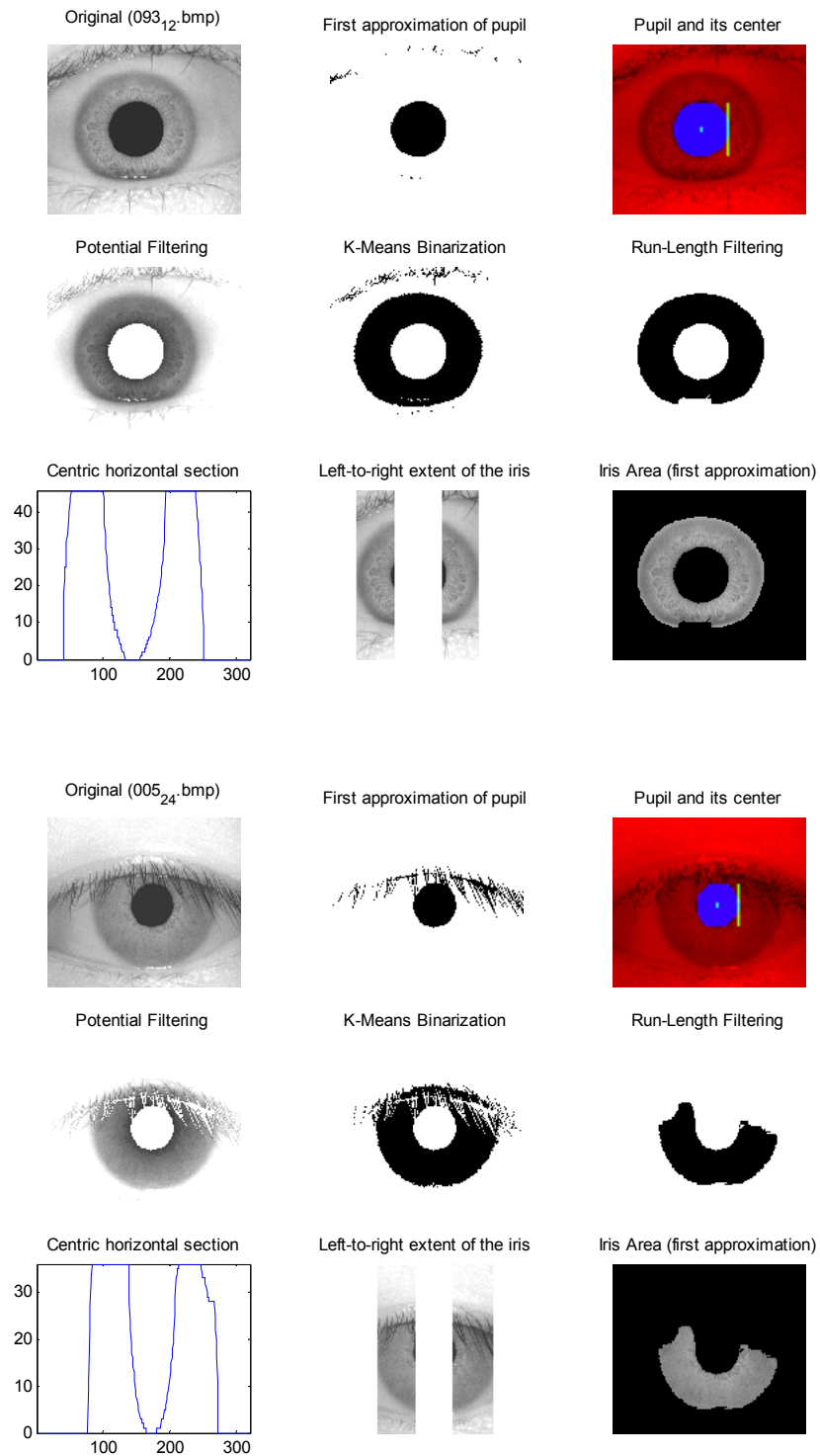


Figure 8 Iris Segmentation: an ideal example / a noisy example.

Our iris segmentation algorithm is the following:

Generic Iris Segmentation Procedure;

1. INPUT: current iris image and eventually other custom data (calibration variables);
2. Run the Fast 4-Means Quantization adaptively to the first cluster consistency to obtain the first approximation of pupil;
3. Use Run-Length Filtering to denoise (eliminate surrounding noise and fill the gaps in the pupil, if any);
4. Identify the pupil with all remaining pixels in the pupil's cluster;
5. Compute the center of the pupil and the pupil radius;
6. Filter entire image with a central potential field of energy originating in the center of the pupil (the iris will become darker and the 'non-iris' whiter).
7. Extract the entire cluster obtained in (2) from the filtered image obtained in (6);
8. Binarize the result of (7) through Fast 2-Means Quantization;
9. Use Run-Length Filtering to denoise (if necessary) the result of (8);
10. If the result from (9) satisfy some consistency criteria then go to (11), else go to (1);
11. OUTPUT: result of (9) is a binary index of a noisy iris segment;

The algorithm obtains very good results but as a whole is still experimental. The pupil localization is fully calibrated for CASIA V1 database and its success rate is 100% on this database. It went wrong (3-pixel error for the pupil center) on a single case which has also being rejected for severe inconsistency of the iris segment (sleepy eye, very noisy eyelashes, pupil severely obstructed by the superior eyelid, all at once).

Few examples of iris segmentation results can be seen in the figure 8.

8. Fast K-Means Image Quantization – an algorithm for image processing supercomputing

This section summarizes some of the most important properties of the main proposed algorithm, Fast K-Means Image Quantization:

First of all, the computational kernel of the algorithm is independent from image size, making the algorithm suitable for very large images.

The algorithm can be used successfully as a component in other applications such as segmentation algorithms.

At the last but not the least we will show that the computational kernel of the proposed algorithm truly does have an incredible speed. The next test has been done over a database of 24-bit RGB usual images containing 84 pieces (from landscape to macro pictures) of dimension 512x512, and is designed to illustrate the time performance of the proposed algorithm. During the test, algorithm (re)quantizes each chroma blue/red plane at the given numbers of chromatic levels mentioned in the Table 1.

Table 1: Time performance of the Fast K-Means Image Quantization Algorithm (Processor: P IV Prescott 2.8 GHz)

Function Name	Fast 128-Means Chroma Quantization				Fast 64-Means Chroma Quantization				Fast 32-Means Chroma Quantization			
	Time(sec.)	Calls	Time/call (sec)	Time(sec.)	Calls	Time/call (sec)	Time(sec.)	Calls	Time/call (sec)	Time(sec.)	Calls	Time/call (sec)
resample_kmeans_uint8	73.90500000	84	0.87982142857141	54.43800000	84	0.64807142857141	36.85300000	84	0.4387261904762			
parse_output	59.71900000	168	0.35547023809520	40.11000000	168	0.238750000000028	22.18300000	84	0.2946547619050			
rgb2ycc_uint8	24.51900000	84	0.29189285714286	24.48200000	84	0.29145238095260	24.75100000	84	0.2810238095239			
ycc2rgb_uint8	23.40900000	84	0.27867857142878	23.34200000	84	0.27788095238070	23.60600000	168	0.1320416666667			
exact_histogram	11.05200000	168	0.06578571428565	11.20100000	168	0.06667261904736	11.22000000	168	0.0667857142857			
imread	4.41800000	84	0.05259523809504	4.47200000	84	0.05323809523816	4.56900000	84	0.0543928571425			
kernel_kmeans_uint8	2.31300000	168	0.01376785714285	2.30000000	168	0.01369047619049	2.75400000	168	0.0163928571428			
recompute_clusters	1.68600000	336	0.00501785714287	1.76800000	627	0.00281977671450	2.17300000	1109	0.0019594229035			
get_new_centroids	0.56300000	336	0.00167559523808	0.45300000	627	0.00072248803830	0.53400000	1109	0.0004815148783			

Function Name	Fast 16-Means Chroma Quantization				Fast 8-Means Chroma Quantization				Fast 4-Means Chroma Quantization			
	Time(sec.)	Calls	Time/call (sec)	Time(sec.)	Calls	Time/call (sec)	Time(sec.)	Calls	Time/call (sec)	Time(sec.)	Calls	Time/call (sec)
resample_kmeans_uint8	27.41000000	84	0.32630952380970	22.47100000	84	0.26751190476183	17.59400000	84	0.2094523809525			
parse_output	11.86300000	168	0.07061309523817	6.55200000	168	0.03900000000026	3.32000000	168	0.0197619047620			
rgb2ycc_uint8	24.60800000	84	0.29295238095217	25.57800000	84	0.30450000000023	25.23400000	84	0.3004047619050			
ycc2rgb_uint8	23.38700000	84	0.27841666666652	24.31400000	84	0.28945238095254	23.91900000	84	0.2847499999998			
exact_histogram	11.13500000	168	0.06627976190464	11.29500000	168	0.06723214285696	11.40900000	168	0.0679107142856			
imread	4.70500000	84	0.05601190476210	4.49900000	84	0.05355952380917	4.72200000	84	0.0562142857140			
kernel_kmeans_uint8	3.71200000	168	0.02209523809532	3.91900000	168	0.02332738095226	2.07800000	168	0.0123690476191			
recompute_clusters	3.01400000	1846	0.00163271939328	3.26100000	2145	0.00152027972027	1.66800000	1479	0.0011277890467			
get_new_centroids	0.52500000	1846	0.00028439869990	0.57900000	2145	0.00026993006993	0.29900000	1479	0.0002021636241			

In the mentioned table we are tracing the trend of the execution times, especially for the computational kernel of the algorithm (kernel_kmeans_uint8), for I/O functions (exact_histogram and parse_output) , and also for 3 ‘witness’ functions: the imread Matlab implicit function, and the RGB-YCC conversion functions (see section 5 - Fast K-Means Image Quantization Algorithm).

It can be seen in the Table 1 that time complexity of the computational kernel is bounded in a narrow band, far away from any polynomial or superpolynomial time [5].

References

- 1 International Telecommunication Union (ITU), The International Telegraph and Telephone Consultative Committee (CCITT), International Standard ISO/IEC, 10918-1: 1993(E), CCITT Recommendation T.81, Digital compression and coding of continuous still images – Requirements and Guidelines, Section 3 (Definitions, abbreviations and symbols), 1993.
- 2 William K. Pratt, Digital Image Processing, Chapter 6 - Image Quantization, John Wiley & Sons, 2001, third edition.
- 3 Weisi Lin, Li Dong, Adaptive Downsampling to Improve Image Compression at Low Bit Rates, IEEE Transactions on Image Processing, vol. 15, no. 9, September 2006 2513.
- 4 Li Ma, Tieniu Tan, Yunhong Wang, Dexin Zhang, Efficient Iris Recognition by Characterizing Key Local Variations, IEEE Transactions on image processing, vol. 13, no. 6, June 2004, 739.
- 5 David Arthur, Sergei Vassilvitskii, How Slow is the k-Means Method?, Stanford University, <http://www.cs.duke.edu/courses/spring07/cps296.2/papers/kMeans-socg.pdf>
- 6 The MathWorks Inc.
<http://www.mathworks.com/support/tech-notes/1100/1109.html>
- 7 Steve Eddins, the MathWorks Inc.,
<http://blogs.mathworks.com/steve/2008/01/28/logical-indexing/>
- 8 <http://www.mathworks.com/matlabcentral/fileexchange/>
- 9 <http://glennchan.info/articles/technical/chroma/chroma1.htm>, figures 6-10,
- 10 <http://www.impulseadventure.com/photo/chroma-subsampling.html>
- 11 http://en.wikipedia.org/wiki/YUV_4:4:4#Types_of_subsampling
- 12 <http://www.sinobiometrics.com/>

Third Party Copyrights

Portions of the research in this paper use the CASIA-IrisV1 database collected by the Chinese Academy of Sciences’ Institute of Automation. Details (including data format, copyright agreement and application procedure) of this database can be found here: <http://www.sinobiometrics.com/english/Databases.asp>