ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

MSc THESIS

Amir POORSADEG ZADEH YEGANEH

COLOR QUANTIZATION WITH NATURE INSPIRED COMPUTING

DEPARTMENT OF COMPUTER ENGINEERING

ADANA, 2012

ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

COLOR QUANTIZATION WITH NATURE INSPIRED COMPUTING

Amir POORSADEG ZADEH YEGANEH

MSc THESIS

DEPARTMENT OF COMPUTER ENGINEERING

We certify that the thesis titled above was reviewed and approved for the award of degree of the Master of Science by the board of jury on 26/09/2012.

.....	.....	.....
Asst. Prof. Dr. Mustafa ORAL	Assoc. Prof. Dr. Mustafa GÜVEN	Assoc. Prof. Dr. Zekeriya TÜFEKÇİ
SUPERVISOR	MEMBER	MEMBER

This MSc Thesis is written at the Computer Engineering Department of Institute of Natural and Applied Sciences of Çukurova University.
Registration Number:

Prof. Dr. Selahattin SERİN
Director
Institute of Natural and Applied Sciences

This study was supported by Scientific Research Projects office of Çukurova University (BAP) under grant No. MMF2011YL18.

ABSTRACT

MSc THESIS

COLOR QUANTIZATION WITH NATURE INSPIRED COMPUTING

Amir POORSADEG ZADEH YEGANEH

ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES
DEPARTMENT OF COMPUTER ENGINEERING

Supervisor : Asst. Prof. Dr. Mustafa ORAL
Year: 2012, Pages: 109
Jury : Asst. Prof. Dr. Mustafa ORAL
: Assoc. Prof. Dr. Mustafa GÜVEN
: Assoc. Prof. Dr. Zekeriya TÜFEKÇİ

In this thesis, two different nature inspired optimization algorithms are applied to the Color Quantization (CQ) problem. CQ is a technique that is used to decrease the number of distinct colors in an image. In many applications, an image with a limited number of colors provides more effective implementations either in software or hardware. However, this process reduces the visual quality of the image. Therefore, one of the major concerns in CQ is keeping the distortion of the quantized image as low as possible. CQ is considered as an NP-Hard optimization problem.

In the first part of this study, for the first time, Intelligent Water Drops (IWD) algorithm is adopted to Basic CQ. The proposed algorithm IWD-CQ was tested on a set of standard test images. The comparison results against eight approaches revealed that IWD-CQ algorithm achieves promising results in terms of the visual quality of quantized images. The algorithms used for comparison include five widely used conventional methods; Uniform, Popularity, Median-Cut, Octree and K-Means and three prominent Artificial Intelligence methods; Self Organizing Maps, Ant Colony Optimization and Simulated Annealing. The results show that none of the mentioned algorithms achieve lower quantization errors than the IWD-CQ algorithm.

Second part of the study, investigates a Perceptual CQ approach which is based on a model of Human Vision System (HVS). The model aims to simulate the property of human's eye in perceiving an average color of the adjacent pixels. The related study was carried out using Differential Evolution (DE) algorithm as the optimizer. Moreover, tuning of the DE parameters for the proposed method is covered. Outcomes of the proposed algorithm, DE-PCQ, are discussed and compared with results of previously mentioned five conventional methods applying the same perceptual approach. Comparison results demonstrate distinct superiority of DE-PCQ in terms of the perceptual quality of quantized images.

Key Words: Color Quantization, Perceptual Color Quantization, Nature Inspired Computing, Intelligent Water Drops Algorithm, Differential Evolution

ÖZ

YÜKSEK LİSANS TEZİ

DOĞA ESİNLİ HESAPLAMA İLE RENK KUANTALAMA

Amir POORSADEG ZADEH YEGANEH

ÇUKUROVA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

Danışman : Asst. Prof. Dr. Mustafa ORAL
Yıl: 2012, Sayfa: 109
Jüri : Asst. Prof. Dr. Mustafa ORAL
: Assoc. Prof. Dr. Mustafa GÜVEN
: Assoc. Prof. Dr. Zekeriya TÜFEKÇİ

Bu tezde, iki doğa esinli optimizasyon algoritması Renk Kuantalama (CQ) problemine uygulanmıştır. CQ, bir resim içerisindeki farklı renklerin sayısını azaltmak için kullanılan bir tekniktir. Sınırlı sayıda renge sahip bir resim, birçok yazılım veya donanımın daha etkin kullanılmasını sağlar. Ancak , bu işlem resmin görsel kalitesini düşürür. Bu yüzden, CQ'daki en büyük kaygılardan birisi kuantalanmış resimdeki bozulmayı mümkün olduğu kadar düşük seviyede tutmaktır. CQ bir NP-Hard problemi olarak değerlendirilmektedir.

Bu çalışmanın ilk kısmında, ilk olarak, Akıllı Su Damlaları (IWD) algoritması temel CQ problemine adapte edilmiştir. Önerilen IWD-CQ algoritması standart test resimleri ile değerlendirilmiştir. Literatürde bulunan sekiz farklı algoritmayla karşılaştırıldığında, IWD-CQ algoritmasının daha iyi sonuçlar elde ettiği görülmüştür. Karşılaştırmalarda sıkça kullanılan beş geleneksel metot (Uniform, Popularity, Median-Cut, Octree, K-Means) ve üç önemli yapay zekâ metodu (Özdüzenleyici Haritalar, Karınca Kolonisi Optimizasyonu ve Tavlama Benzetimi) kullanmıştır. Sonuçlar, IWD-CQ algoritmasının diğer algoritmalara göre daha az kuanlatama hatası ile çalıştığını göstermektedir.

Çalışmanın ikinci kısmında, İnsan Görüş Sistemine (HVS) dayalı, algısal CQ algoritma modeli incelemektedir. Bu model, insan gözünün komşu piksellerin renk ortalamasını algılama özelliğinin, bir benzetimidir. Optimizasyon yöntemi olarak Diferansiyel Gelişim (DE) algoritması kullanılmış ve parametrelerinin nasıl ayarlandığı anlatılmıştır. Önerilen DE-PCQ algoritmasından elde edilen sonuçlar, aynı algısal modelin daha önce bahsedilen beş geleneksel metoda uygulanması sonucu elde edilen sonuçlar ile karşılaştırılmış ve tartışılmıştır. Karşılaştırma sonuçları, DE-PCQ algoritması ile quantalanan resimlerin algısal kalitesinin bariz bir şekilde daha üstün olduğunu göstermektedir.

Anahtar Kelimeler: Renk Kuantalama, Algısal Renk Kuantalama, Doğa Esinli Hesaplama, Akıllı Su Damlaları algoritması, Diferansiyel Gelişim algoritması

ACKNOWLEDGEMENTS

First of all I would like to express my utmost gratitude to the merciful divinity who granted us all the beauties and blessings of the existence and provided us the power and insight to overcome difficulties and to obtain achievements.

My very special thanks go to my parents whom I am deeply indebted for all the things they gave me in my life including the opportunity to study. They always supported and stimulated me to go on. My appreciation to my parents for their endless support and irrecoverable sacrifice cannot be described by words.

It is my great pleasure to thank my advisor Asst. Prof. Dr. Mustafa ORAL. I cannot overstate his valuable contributions and discussions on this study, as well as many other subjects of interest which extended my knowledge. I greatly appreciate his contribution, inspiration, and friendship and I hold him in high esteem on both the professional level and the personal level.

I also would like to thank the members of the jury committee: Assoc. Prof. Dr. Zekeriya TÜFEKÇİ and Assoc. Prof. Dr. Mustafa GÜVEN for their consideration of the study and their valuable feedbacks at the oral examination. Finally, I thank all those who helped me for this achievement especially Dr. Ismail AZAD for his kind help, support and advises during my undergraduate studies and afterwards.

Amir POORSADEG ZADEH YEGANEH

CONTENTS **PAGE**

LSIT OF TABLES **PAGE**

LSIT OF FIGURES **PAGE**

LIST OF ABBREVIATONS

CQ	: Color Quantization
PCQ	: Perceptual Color Quantization
U-CQ	: Uniform Color Quantization
P-CQ	: Popularity Color Quantization
MC-CQ	: Median Cut Color Quantization
O-CQ	: Octree Color Quantization
Km-CQ	: K-means Color Quantization
JPEG	: Joint Photographic Experts Group
GIF	: Graphic Interchange Format
NP-Hard	: Non-deterministic Polynomial-time Hard
IWD	: Intelligent Water Drops
GA	: Genetic Algorithm
SA	: Simulated Annealing
CMA	: C-means Clustering Algorithm
ANN	: Artificial Neural Networks
NN	: Neural Networks
ACR	: Adaptive Color Reduction
NNC	: Neural Network Classifier
PCA	: Principal Component Analyzer
PCC	: Principal Color Components
SOFM	: Self Organized Feature Map
SONNC	: Self Organized Neural Network Classifier
VLSI	: Very Large Scale Integration
HBMO	: Honey Bee Mating Optimization
PSO	: Particle Swarm Optimization
GLA	: Generalized Lloyd Algorithm
MAE	: Mean Absolute Error
MSE	: Mean Square Error
FWMSE	: Frequency-Weighted Mean Square Error

MKP : Multiple Knapsack Problem

TSP : Travelling Salesman Problem

AI : Artificial Intelligence

LBG : Linde–Buzo–Gray Algorithm

HVS : Human Visual System

TSVQ : Tree Structured Vector Quantizer

PGF : Peer Group Filtering

IIF : Inverse Image Frequency

CIE : International Commission on Illumination (French *Commission internationale de l'éclairage*)

1. INTRODUCTION

1.1. What is Color Quantization (CQ)?

Color Quantization (Heckbert, 1982) is the process of selecting a limited number of colors to represent an image which has a large number of distinct colors. Color Quantization (CQ) is comprised of two phases; palette selection, and color mapping. In the first phase, it is aimed to select an appropriate subset of colors such that the quantized image is represented with the highest similarity to the original one. In the second phase, color of each pixel from the original image is mapped to its best matching color from the palette. As a result of CQ, the image is reconstructed using less data and less distinct colors.

Color quantization of the images is a lossy process. In other words, a portion of the image data is eliminated during the process and consequently some distortion appears in the quantized image. Therefore, one of the main concerns of CQ is to keep the visual artifacts of the quantized image as low as possible.

1.2. Applications of CQ

Initial aim of the CQ is to display images with many colors on devices that can only display a limited number of colors. Because of hardware limitations, some display devices are not able to represent all the possible colors simultaneously. Therefore, the number of colors in a given image should be decreased properly before the image is shown on the display. Although most of today's display and printer devices are able to represent true-color images easily, CQ has not lost its importance. The true-color images employ 24 bits per pixel to represent the colors as same as they appear in real life to human's eye. The mentioned color depth provides approximately 16.7 million possible colors for each pixel. Processing, storage and transmission of this amount of data is problematic for many applications and causes inefficiency. Reducing the number of distinct colors improves the efficiency of system by decreasing the amount of data that the system should deal with.

Furthermore, CQ is a necessity in systems that cannot work with true-color images due to hardware or resource limitations. Some examples for applications of CQ include;

- Display and print devices: CQ is extensively used to optimize the display and print quality of the images on devices that are not able to display or print true-color images.
- Web pages: A considerable amount of internet bandwidth is used to transfer the images used in web pages. Decreasing the amount of image data saves the internet bandwidth and results in faster loading of the web pages.
- Online video streaming: A video stream is simply a sequence of images (frames) that are displayed sequentially. Reducing the size of each frame will reduce the required bandwidth in an online video streaming application.
- Image segmentation: CQ is often used as a preprocessing stage in image segmentation to simplify the images.
- Image storage: In some applications (e.g. Medical Imaging), it is not needed to store the acquired images with detailed color information. In such situations, the extra color details can be eliminated by a color quantization process.
- Type conversion: Reducing the number of colors is a necessity in some type conversions. For instance, a conversion from JPEG to GIF requires that the number of colors be reduced to maximum 256 colors.

1.3. Complexity of CQ problem

Color Quantization is a complex data clustering problem because of its multi-dimensional solution space. More specifically, CQ is a Non-deterministic Polynomial-time hard (NP-hard) optimization problem (Heckbert, 1982; Brucker, 1977).

As previously mentioned, the aim of CQ is to select a subset of colors such that the final image is represented in the best way. The subset is usually selected from the set of colors in the given image. Regarding that the order of selected colors does not matter and the repetition is not allowed, CQ can be considered as a combinatorial problem of selecting k distinct items (colors) out of n items. The number of possible combinations is denoted by $\binom{n}{k}$ and is calculated as follows (binomial coefficient).

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!} \tag{1.1.}$$

The outcome of Eq. 1.1 may be an extremely large number. For instance, we may consider the quantization of Peppers image in Fig. 1.1.a. to eight colors. Regarding that the image contains 183,525 distinct colors, the number of possible combinations is calculated as follows:

$$\binom{183{,}525}{8} = \frac{183{,}525!}{8! \times 183{,}517!}$$

$$= 31{,}913{,}547{,}158{,}371{,}401{,}925{,}804{,}095{,}793{,}613{,}642{,}700$$

Each combination is a candidate color palette for the optimal solution of the problem. If it is assumed that the evaluation of each combination takes only one millionth of a second, the evaluation of all the possibilities will take 10^{24} years! The quantization result obtained from one of the above combinations is shown in Fig. 1.1.b. The correspondent combination is selected by uniform color quantization method. In the case of quantizing the same image to 256 colors, the number of possible combinations will be approximately 3.13×10^{840} which will take 10^{827} years to be evaluated!

a.  b. 

Figure 1.1. Sample original and quantized images; a) A typical true-color image; Peppers; b) Uniform quantization of the image to eight colors.

1.4. CQ with Nature Inspired Computing

The importance of Color Quantization has been the motivation of many researches to investigate better approaches to the problem. In this thesis, two nature inspired optimization algorithms are studied in this regard.

1. For the first time, Intelligent Water Drops (IWD) Algorithm is adapted to the problem in order to generate optimal or near optimal palettes for gray-level quantization of the images.
2. Differential Evolution (DE) algorithm is applied to the problem based on a model of Human Visual System with the aim of increasing the perceptual quality of the quantized images.

Although computational cost of an algorithm is an important issue, this study does not investigate minimizing the computational cost of the studied algorithms but improving the final quality of the quantized images. Therefore, the main focus of the study is to develop more effective algorithms in order to find optimal or near optimal palettes in color quantization. The performances of aforementioned two approaches are compared with the performances of several conventional and artificial intelligence based methods for CQ.

1.5. The Outline of Thesis

This study is organized as follows; Section 1 provides introductory information about the Color Quantization and the thesis (current section). Section 2 presents previous works on CQ in three groups; Conventional methods, AI based methods, and Perceptual CQ methods. Section 3 discusses CQ in more details and describes standard versions of IWD and DE algorithms. Rest of the Section 3 presents the proposed CQ approaches; IWD-CQ and DE-PCQ. Results and Discussions are provided in Section 4. Conclusions are summarized and presented in Section 5.

2. PREVIOUS STUDIES

The importance of Color Quantization and need for efficient methods in this regard, have been the motivation of numerous researches to develop different algorithms. Considering the structure of the present study, the previous works are categorized in three groups; Conventional CQ methods, AI based CQ methods and Perceptual CQ methods.

2.1. Conventional CQ Algorithms

Uniform CQ (Heckbert, 1982; Hill, 1990) (U-CQ) is the simplest and the fastest conventional method which simply divides each color axis into an arbitrary number of sub-ranges. Each axis is treated independently and perpendicular planes crossing from the division points divide the color space into equal volume regions. Each region provides a representative color for all of the colors that fall into that region. This implementation is fast and does not require a first pass over the image to generate the partitioning scheme. However, a partitioning that does not consider color distribution of the images, generally, is not promising to give optimal results (Heckbert, 1982). The main disadvantage of U-CQ is wasting of colors in the regions that do not have any pixels to be represented, whereas in another region, a multitude number of colors may be represented by only one color. Nevertheless, effective implementations of U-CQ which yield comparable results with other methods are still achievable (Mojsilovic, 2001).

In order to improve the quality of resulting image, quantization methods such as Octree CQ (Gervautz and Purgathofer, 1990; Clark, 1996; Bloomberg, 2008a), Median-Cut CQ (Kruger, 1994; Bloomberg, 2008b), Popularity CQ (Clark, 1995) and utilize color statistics information of the image.

The principle of the Octree Color Quantization (O-CQ) is to read the image pixels and place their colors in an Octree of depth 8. This method provides fast indexing for an inverse color table. Every leaf at depth 8 represents a distinct color and the tree has a limit of K leaves (ex. K = 256). One implementation for O-CQ

(Gervautz and Purgathofer, 1990) builds the Octree by taking pixels, one at a time, and either making a new color or merging them with an existing color. Merging process results in an effective pruning of the tree so only up to K colors need to be stored at any time. On the other hand different possibilities for merging process, makes the algorithm complicated.

The Median-Cut CQ (MC-CQ) algorithm repeatedly divides the longest sub-range of the color distribution, trying to put roughly equal numbers of pixels in each of two resulting regions. The sub-color space is divided each time by a plane perpendicular to one of the color axes (the longest one) from its median point. This process is repeated for every newly generated region until the desired number of regions is obtained. Therefore MC-CQ algorithm aims to create color regions containing equal number of colors. This characteristic turns into a disadvantage in the areas with a low density color distribution, because in such areas, the volume of the constructed regions will be very large and this results in large quantization errors in those regions (Bloomberg, 2008a). Finally, the resulting palette is obtained from the centroids of each region.

The Popularity CQ (P-CQ) constructs the palette by selecting the colors from the set of most popular colors. Before determination of the more repeated colors, depth of the color space is reduced to make a reasonable spacing between adjacent colors. Then, N more repeated colors are selected to represent the original image. This method is not effective when the image owns many different colors and it may cause a small set of pixels of some particular colors never be represented by a nearby appropriate color, therefore their colors will be lost (Bloomberg, 2008a).

Clustering methods are also adoptable to the CQ problem. K-means as a well-known unsupervised clustering algorithm is widely used in Color Quantization (Verevka and Buchanan, 1995; Kasuga et al., 2000; Celebi, 2009; Celebi, 2011). A general implementation of K-means algorithm for CQ (Km-CQ) starts with an initial set of K centers which K is the number of colors in the palette. Then, each pixel is assigned to the nearest center and clusters are formed. In the next step, for each cluster, its new center is computed and these centers are used as the centers of the next run of the algorithm. This process is repeated until some stop criterion is

satisfied, for example until no movement in the centroids is detected. The final result of K-means is highly dependent on the initial set of centers and the algorithm may easily get stuck in a local optimum. Therefore some researches aim to find an effective initialization for K-means (Celebi, 2009) or to improve the performance of the algorithm (Celebi, 2011).

2.2. Artificial Intelligence CQ Algorithms

In addition to conventional methods described in Section 2.1, a wide variety of modern Artificial Intelligence based approaches have been applied to CQ problem.

Simulated Annealing (SA) is a probabilistic metaheuristic for the global optimization problems. It is inspired from annealing process in metallurgy and is able to approximate global optimum of a given function in a large search space. Vaisey and Gersho (1988) aimed to alter Generalized Lloyd Algorithm (GLA) with techniques motivated by SA in order to improve the quality of the codebooks by reducing the distortion involved in coding the training set.

Dekker (1994) applies Self-Organizing Kohonen Neural Network for quantizing color graphics images. It has been shown experimentally that, by adjusting a quality factor, the network can produce images of much greater quality with longer running times, or slightly better quality with shorter running times than the existing methods.

Genetic Algorithm is a well-known stochastic optimization approach which has been studied extensively in the literature. A hybrid approach combining genetic algorithm with classical C-Means clustering Algorithm (CMA) was presented by Scheunders (1997). The proposed technique was superior to CMA in the sense that it converged to a nearby global optimum rather than a local one.

Zheng et al. (1997) elaborated color quantization in the context of vector quantization which maps groups of input symbols, called vectors, onto a small set of vectors, called the codebook. They described a genetic algorithm for the problem of codebook design. To speed up the operations, they used fitness inheritance to assign

fitness values to the most of the new chromosomes, rather than evaluating them. They compared the proposed algorithm to a popular non-genetic algorithm for codebook design. The genetic algorithm was found to be effective, but slow.

Freisleben and Schrader (1997a) proposed an adaptive version of hybrid GA/K-means algorithm to solve the CQ problem. The proposed algorithm was adaptive in the sense that a given image is sub-sampled with varying sampling rates during the execution of the hybrid GA. The feasibility of the approach was demonstrated by presenting results for some test images. The results indicate that their proposal was superior to the other post-clustering approaches in terms of both computation time and image quality.

Evolutionary Algorithm (EA) is a generic population-based metaheuristic optimization algorithm. EA uses some mechanisms inspired by biological evolution: reproduction, mutation, recombination, and selection. Freisleben and Schrader (1997b) proposed a new post-clustering approach to the color quantization problem which is based on the combination of EA and K-means approach. In the proposed algorithm K-means is used to locally optimize the individuals of the subsequent generation once it had been formed by appropriate evolutionary operators.

Taşdizen et al. (1998) discussed the nature and the difficulty of the color quantization problem and its formulation. They applied GA to CQ problem and studied the effect of the parameters such as mutation and crossover probabilities and population size on the quality of solutions. In their work, solutions obtained with genetic algorithm were compared with those of heuristics and the K-means clustering algorithm. They reported that the results produced by GA were in superior to the other approaches tested.

Papamarkos (1999) proposed an approach based on Kohonen self-organized feature map neural network. In the proposed algorithm, after training, the neurons of the output competition layer define the proper color classes. The final image has the dominant image colors and its texture approaches the image local characteristics used.

Franti (2000) applied GA to CQ problem and s/he introduced a crossover method as the most important factor of the algorithm. In the mentioned study, a new

deterministic crossover method based on the pairwise nearest neighbor method is proposed. s/he showed that high quality codebooks can be obtained within a few minutes instead of several hours as required by the previous GA-based methods. The method outperformed all comparative codebook generation methods in quality for the tested training sets.

Gonzalez et al. (2000) introduced an Evolution-based Adaptive Strategy for Color Quantization of sequences of images. In the proposed algorithm individuals correspond to individual cluster centers. In order to approach real-time response one-generation adaptation was imposed for each image. Mutation operators were guided by the actual covariance matrices of the clusters. Experimental results on a sequence of indoor images were presented in the study.

Puzicha et al. (2000) developed a method based on Deterministic Annealing to simultaneously quantize and halftone color images. The method is based on a rigorous cost–function approach which optimizes a quality criterion derived from a simplified model of human perception. It incorporates spatial and contextual information into the color quantization process.

Özdemir and Akarun (2002) employed a Fuzzy Logic approach in their solution to CQ problem. They proposed a new fuzzy color quantization technique which incorporates a term for partition index. The partition index term is used to modify the membership function such that higher membership values are assigned to nearby vectors with respect to codebook entries and lower membership values are assigned to further vectors. The overall effect of the inclusion of the partition index is not only to obtain quantized images with lower MSE (Mean Square Error), but also with lower FWMSE (Frequency-Weighted Mean Square Error) both in luminance and chrominance channels.

Atsalakis et al. (2002a) introduced a technique that splits the image into windows and uses Kohonen Self Organized Neural Network Classifier (SONNC). Initially, the dominant colors of each window are extracted through the SONNC and then are used for the quantization of the colors of the entire image. Moreover, they introduced VLSI implementation of the proposed algorithm. In addition, Atsalakis et al. (2002b) proposed another CQ technique which is based on the use of the image

Principal Color Components (PCC). PCC includes the image color components and additional image components extracted with the use of proper spatial features. Using Kohonen SOFM as classifiers, the principal color components of each PCC are obtained and a look-up table, containing the principal colors of the PCC, is constructed. The final colors are extracted from the look-up table entries through a Kohonen SOFM by setting the number of output neurons equal to the number of the principal colors obtained for the original image.

Artificial Neural Network (ANN) or simply Neural Network (NN) which is an interconnected group of nodes was inspired from biological neural networks. It is usually used to model complex relationships between inputs and outputs or to find patterns in data. In the literature, different NN based approaches has been applied to CQ problem. Papamarkos et al. (2002) developed an Adaptive Color Reduction (ACR) technique which achieves color reduction using a tree clustering procedure. In each node of the tree, a self-organized Neural Network Classifier (NNC) is used which is fed by image color values and additional local spatial features. The NNC consists of a Principal Component Analyzer (PCA) and a Kohonen Self-Organized Feature Map (SOFM) Neural Network. The output neurons of the NNC define the color classes for each node. The final image not only has the dominant image colors, but also its texture approaches the image local characteristics used.

Ant Optimization is a probabilistic technique for solving optimization problems. It is inspired from the behavior of ants seeking a path between their colony and a source of food. This optimization approach has been applied to color quantization by Xinrong et al. (2005). The focus of the study was on grouping similar data objects of a color image into a cluster. The dominant color of a cluster comes from the colors with more pixels in a color image so that it is able to avoid image distorting produced by executing traditional quantization algorithm.

Particle Swarm Optimization (PSO) was a population-based optimization algorithm that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. A color image quantization algorithm based on PSO was developed by Omran et al. (2005). Each particle in the swarm is randomly initialized to contain K centroids (i.e. color triplets). The K-

means clustering algorithm is applied to each particle at a user-specified probability to refine the chosen centroids. Each pixel is then assigned to the cluster with the closest centroid. PSO is used to refine the centroids obtained from the K-means algorithm.

Schaefer and Nolle (2006) applied a variant of SA as a standard black-box optimization algorithm to the color quantization problem. The main advantage of black-box optimization algorithms is that they do not require any domain specific knowledge yet are able to provide a near optimal solution.

Differential Evolution is an optimization technique that has been successfully employed in various applications. Schaefer and Nolle (2006) have applied Differential Evolution to the problem of generating an optimal color map for color quantized images. They showed that Differential Evolution can be effectively employed as a method for choosing the entries in the map. In order to optimize the image quality, they combined Differential Evolution approach with a local search method to guarantee finding the local optimal color map. This hybrid approach outperformed various commonly used color quantization algorithms on a set of standard images.

El-Mihoub et al. (2006) showed that a self-adaptive hybrid genetic algorithm can be employed to generate a palette of high quality. Their proposed method employed a novel local search method and outperformed several conventional color quantization algorithms providing superior image quality.

Pan and Cheng (2007) presented an evolution-based tabu search approach to design codebooks. In the ETSA, there is no need for users to determine the size of a tabu memory and to specifically define a set of tabu restrictions and a set of aspiration criteria. During the iterations, only the best solution visited is memorized as a tabu point in the search space and the distance from each trial solution to the tabu point is an important factor in the fitness evaluation. In population competition, the new fitness function plays the roles of the tabu restrictions and the aspiration criteria. Based on the new fitness function and a parallel evolutionary mechanism, the ETSA can prevent premature convergence and eventually find a good solution.

Honey Bee Optimization is a population-based search algorithm which mimics the food foraging behavior of swarms of honey bees. Horng and Jiang (2011) applied this algorithm to construct the codebook of vector quantization. The study gives a detailed description of how the HBMO (Honey Bee Mating Optimization) algorithm can be used to implement the vector quantization and enhance the performance of Linde–Buzo–Gray (LBG) algorithm for vector quantization.

2.3. Perceptual CQ Algorithms

Atkins et al. (1994) introduce a frame based color image quantization method which accounts for the spatial and temporal frequency dependence of the contrast sensitivity of the luminance and chrominance channels. The method extends an existing technique for design of an optimal filter to the temporal dimension. It is observed that better image quality is yielded compared with frame-independent methods. However, a sufficient high frame rate is needed for human visual system to support the temporal averaging.

Human visual system has higher sensitivity to quantization errors in low frequency regions of an image compared to errors in high frequency regions. This property is employed by Chaddha et al. (1994) to develop a subjective distortion measure for color quantization. In their study, the mentioned measure is applied to a cubic pre-quantizer and to a weighted Tree Structured Vector Quantizer (TSVQ) to obtain the final 8-bit color palette. It is shown that the algorithm exhibits higher visual quality than those methods of dithering and error diffusion.

The algorithm proposed by Deng et al. (1999), also, employs the property of human visual system in being less sensitive to the quantization errors in high frequency areas. Their method uses a Peer Group Filtering (PGF) as a preprocessing step to smooth and remove impulse noises from the images. Then, the local statistics gathered from the PGF is used to decrease the number of clusters in detailed areas of the image. This method helps to preserve the important information of the image while eliminating the information which is unimportant from the human's point of view.

Puzicha et al. (2000) propose a method to simultaneously quantize and halftone the images. The method employs a rigorous cost function to optimize a quality criterion which is based on a simplified model of human perception. The capability of human visual system to distinguish different colors is low for high frequency areas. Moreover, a spatial averaging is performed in visual system of the human. The study takes into account spatial and contextual information of the images to combine the quantization and half-toning. A significant improvement in the image quality of final images is reported.

Yoon and Kweon (2004) propose a perceptual based CQ algorithm by constructing two kinds of maps; homogeneity map (H-map) and distinctiveness map (D-map) which are obtained from analyzing the spatial color distribution of the image. These maps are then combined by means of assigning weight values to all color vectors. A new cost function along with local K-Means algorithm is employed to extract representative colors. An incremental splitting scheme is used to adaptively determine the optimal number of clusters. The proposed algorithm is shown to preserve the significant local features of the image while removing unimportant details from the viewpoint of human eye.

Shah et al. (2004) utilize the information of color variations in different regions of the image along with histogram information of the image to devise an effective approach for perceptual color quantization. Moreover, Inverse Image Frequency (IIF) is introduced in order to compute the representative colors of the image. The proposed algorithm is compared with median cut algorithm and its superiority to the median cut algorithm is shown.

Kwak et al. (2005) take into account the area of spatial sensitivity in their algorithm as an important characteristic of the HVS. It is based on the variations of the primary colors in an image. The proposed algorithm employs an improved binary tree vector quantization along with a weight factor which is obtained from the mentioned spatial activity. The weight factor affects the process of splitting the nodes. The study shows that the algorithm yields better image quality and higher PSNR values than conventional methods.

Based on the fact that final result of the CQ process will be evaluated by human's perception, some CQ approaches aim to consider different properties of the Human Visual System to increase the perceptual quality of the final images. Rest of the section presents several approaches proposed in this regard.

Zhou et al. (2008) proposed a perceptual quantization method which is based on a Back Propagation Neural Network approach. Their research is based on the studies from various fields of science emphasizing the importance of 9-13 colors in human color perception. Zhou et al. consider 11 focal colors: black, white, red, green, yellow, blue, brown, purple, pink, orange and gray under the assumption that firstly they are robust to variability among people and secondly they are robust to variability within individual observers. Their proposed method introduces a color quantization scheme which takes into account human perception according to human vision habits.

Lixia (2009) introduces a human perception based CQ algorithm which aims to minimize the distortion in quantized image with respect to the perceptually uniform Munsell NBS distance measurements. The proposed method forms color clusters by choosing the most frequently used colors as seeds and puts limitation of the optimal distance between them. The method is compared with Minimal Variance Quantization (MVQ) algorithm and its superiority is shown in terms of quantization distortion and perceptual effect.

3. MATERIAL AND METHOD

3.1. Material

3.1.1. Color Quantization

Color Quantization is a widely used method to reduce the amount of data that is used to represent an image. However, decreasing the amount of data causes artifacts to appear in the image. Thus, minimizing the visual artifacts is an important concern in CQ. This section introduces the concept of basic Color Quantization and describes its drawbacks. Rest of the section introduces a halftoning method which is used to decrease the artifacts by taking into account certain properties of the Human Visual System.

3.1.1.1. Basic CQ

The goal of CQ methods is finding a color palette that minimizes the quantization error. A palette contains the colors that are used to represent a quantized image. Numerous algorithms have been proposed with the aim of finding the best possible color palette. Once the palette is determined, a mapping method is used to replace color of each pixel with its best matching color from the palette. Basic color quantization does not introduce any other mechanism to improve the visual quality of the quantized images. Therefore, the main consideration of basic CQ is to minimize the quantization error caused by replacing the colors of the original image with the colors included in the palette.

Different CQ algorithms cause different artifacts in the quantized images. One of the most common distortions is the appearing of Mach bands especially in the areas having smooth transition between colors (gradient areas). Figure 3.1 illustrates this artifact which is caused by a Uniform CQ algorithm.

a.  b. 

Figure 3.1. Uniform Color Quantization; a) Original image represented with 24 bits/pixel (8 bits/channel); b) Quantized image represented with 8 bits/pixel (R=3, G=3, B=2).

Losing the less popular but important colors in an image is another artifact that commonly is caused by the Popularity CQ algorithm. Regarding that the Popularity algorithm aims to minimize the quantization error by choosing the most popular colors, less popular colors lose their chance to be represented with an appropriate color. This artifact is illustrated in Figure 3.2.

a.  b. 

Figure 3.2. Popularity Color Quantization; a) Original image; b) Quantization result with the Popularity algorithm. It can be seen that the blue color which represents an important component of the image is lost.

Other commonly used CQ algorithms discussed in this thesis including Median-Cut, Octree, and K-Means, generally perform better than Uniform and Popularity algorithms. Nevertheless, they are still vulnerable against special situations that may produce significant artifacts being discussed in Section 2.1. Basic CQ only considers developing better algorithms in order to find the best possible palette. However, other complementary methods can be used along with Basic CQ in order to overcome its drawbacks (e.g. Halftoning).

3.1.1.2. Error Diffusion

Error diffusion is a halftoning technique that is used to increase the visual quality of the quantized images. Halftoning is a method that aims to simulate continuous tone using dots varying either in size, in shape or in spacing.

Figure 3.3a demonstrates a gray-scale image with continuous gradient. Figure 3.3b is a halftoned image that simulates the gradient effect of the original image by using only black dots on a white plane. Here, size of the dots controls the effect.

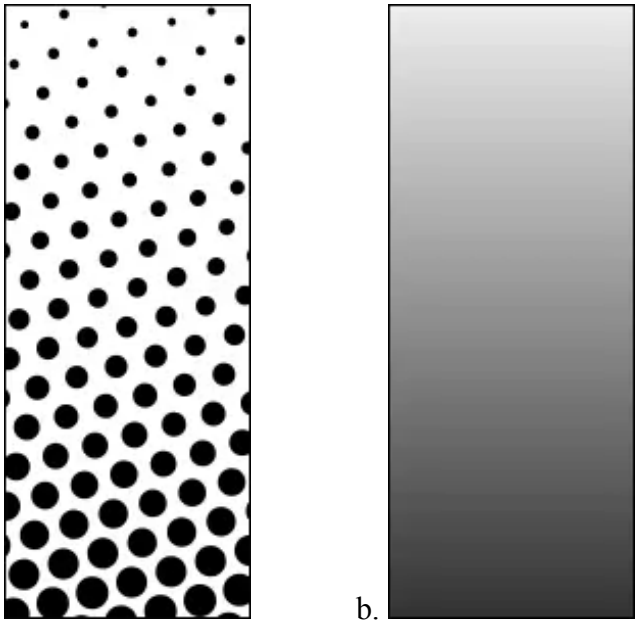


Figure 3.3. Halftoning by Halftone dots; a) A gray-scale gradient; b) Simulation of the gradient using halftone dots. The left image appears same as the right one from a sufficient distance.

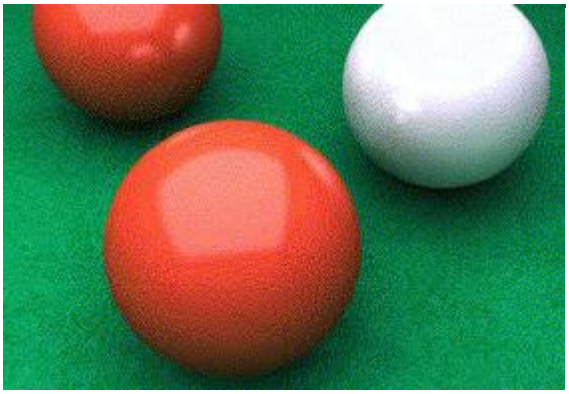
Error diffusion is an intentionally applied form of noise used to randomize quantization error. In this method, quantization error of each pixel is immediately diffused to its neighboring pixels. This process simulates the gradient colors when the image is seen from a sufficient distance. The Mach bands artifact can be removed effectively by using this method.

The effectiveness of error diffusion is illustrated in the following figure. Figure 3.4a demonstrates a common RGB image having smooth color transitions. Figure 3.4b shows the same image being quantized to 32 colors using uniform CQ

method. In Figure 3.4c the same configuration is applied with error diffusion. It can be seen that the quality of final result is improved significantly.

a) Original image.

b) Uniform quantization to 32 colors (R=2 bits, G=2 bits, B=1 bit).



c) Uniform quantization to 32 colors with error diffusion.

Figure 3.4. Error diffusion Halftoning; The final perceived quality of the image is improved dramatically.

Different error diffusion methods are available in the literature including Floyd–Steinberg dithering (Floyd and Steinberg, 1976), Stucki dithering, Burkes dithering and etc. (Lee Daniel et al., 1991). In this study, Stucki method is applied which is based on the kernel presented in Figure 3.5:

		*	8/42	4/42
2/42	4/42	8/42	4/42	2/42
1/42	2/42	4/42	2/42	1/42

Figure 3.5. Kernel for Stucki error diffusion method. * indicates the quantized pixel.

Kernel determines the ratios of quantization error that will be added to each neighboring pixel. It should be noted that after quantization of each pixel, the quantization error is immediately diffused to the neighboring pixels based on the given ratios.

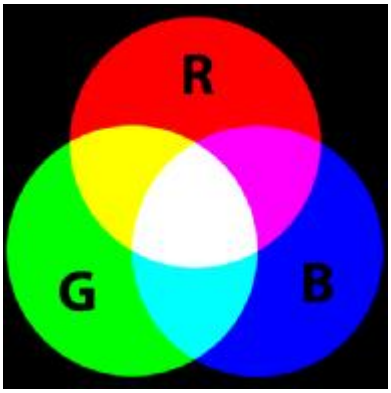
3.1.2. Color Spaces

Colors in the real world are perceived by three types of receptor cells which are placed in the retina. These receptors, called cones, measure a part of the electromagnetic spectrum, approximately between 300 nm and 830 nm (Tkalcic and Tasic, 2003). However, some combinations of this spectrum are not visible to us due to certain characteristics of the Human Visual System (HVS). Different cones have different spectral sensitivities which are grouped into L, M, and S for **L**ong, **M**edium, and **S**hortwave sensitivity. A "Color" is the human perception from a visible part of the mentioned electromagnetic spectrum and "Color Space" is a notation by which the colors are specified. Several color spaces are available in the literature (Poynton, 1995; Palus, 1998; Wyszecki and Stiles, 2000) including RGB, CIEXYZ, CIELuv, CIELab, CMYK, HSV, HSL and etc. Each color space possesses certain properties which make it appropriate for certain type of applications.

3.1.2.1. RGB Color Space

Based on early studies about color science (Wright, 1981), it is known that the mentioned three types of receptors are approximately sensitive to Red, Green and Blue colors. This was the motivation behind RGB color space which describes the colors by using three major color components; Red, Green, and Blue. RGB colors space uses additive color mixing. In other words, colors are obtained adding different

intensities of Red, Green and Blue colors together. Figure 3.6 illustrates how different colors are obtained by mixing the three major colors. The mixture of three components in their maximum intensities produces white color.

a. 

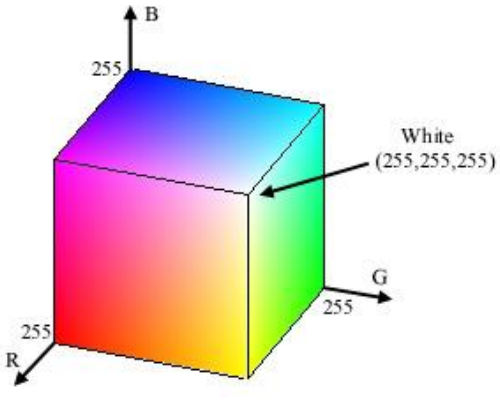
b. 

Figure 3.6. a) Additive color mixing in RGB color space.
b) RGB Color Cube

In the most widely used implementation of the RGB color space, each color component is represented with 8 bits. Thus, 24 bits are used to determine the color of each pixel. Moreover, some implementation use another 8 bits called alpha channel to specify the transparency of a pixel. Assuming that the sensitivity of receptors are defined by functions $R(\lambda)$, $G(\lambda)$, and $B(\lambda)$, the value of R, G and B components are calculated as follows:

$$R = \int_{300}^{830} S(\lambda)R(\lambda)d\lambda$$

$$G = \int_{300}^{830} S(\lambda)G(\lambda)d\lambda$$

$$B = \int_{300}^{830} S(\lambda)B(\lambda)d\lambda$$

where $S(\lambda)$ represents the light spectrum. In other words, the value of R, G and B components are the sum of the respective sensitivity functions and the incoming light. Although RGB color space has some shortcomings like the low correlation between the perceived difference of two colors and Euclidian distance in

the RGB space, it is widely used in display devices and in the literature of image processing.

3.1.2.2. CIELab Color Space

In 1976 CIELuv and CIELab color spaces are proposed by the CIE (International Commission on Illumination) with the main goal of introducing a perceptually uniform color space. They were derived from the earlier color space CIEXYZ which its major property is being device independent. Perceptual uniformity in CIELuv and CIELab color spaces implies that Euclidian distances between colors are strongly correlated with the human visual perception. This property is based on two constraints; chromatic adaptation, and non-linear visual response. Figure 3.7 presents an illustration of the CIELab color space.

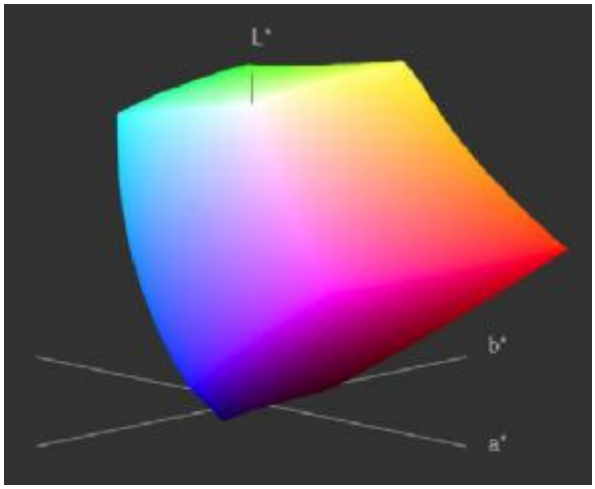


Figure 3.7. CIELab Color Space.

A CIELab color is composed of three components L^*, a^*, and b^*. Range of L^* component varies between 0 and 100 which corresponds to black and white respectively. The possible range of a^* and b^* components is independent of the color space that is being converted from. Nonetheless, the ranges [-110,110] or [-128,128] is possible for a^* and b^* components.

A perceptually uniform color space provides a better framework for implementing perceptual algorithms. Therefore, CIELab color space is used in Section 3.2.2 to implement the proposed algorithm; Perceptual CQ with DE.

3.1.3. Intelligent Water Drops Algorithm

3.1.3.1. Overview

The nature has always been a unique and valuable source of inspiration for scientists to develop new approaches for problems in different fields of science and technology. Genetic Algorithms (Holland, 1975), Ant Colony Optimization (Colorni et al, 1991; Dorigo, 1992), Bees Algorithm (Pham et al., 2005; Pham et al., 2006), River Formation Dynamics (Rabanal et al., 2007), Particle Swarm Optimization (Kennedy and Eberhart, 1995; Shi and Eberhart, 1998), Galaxy-based Search Algorithm (Shah-Hosseini, 2011a; Shah-Hosseini, 2011b), Gravitational Search Algorithm (Rashedi et al., 2009), Charged System Search (Kaveh and Talatahari, 2010), Firefly Algorithm (Yang, 2008), Glowworm Swarm Optimization (Krishnanand and Ghose, 2005), Invasive Weed Optimization (Mehrabian et al., 2006) and Cuckoo search (Yang, 2009; Yang and Deb, 2010) are among the algorithms which are inspired from the nature. The Intelligent Water Drops (IWD) Algorithm is another swarm-based algorithm for optimization problems (Shah-Hosseini, 2007) which is adapted to CQ problem in this study.

IWD algorithm is inspired from the processes that happen between water drops and riverbed, while the drops are flowing in the river. Based on the observation of the nature, it is known that in spite of the obstacles of the environment, a river can find its optimum or near optimum path from the source to its destination. This algorithm is applied to a variety of problems; Traveling Salesperson Problem (TSP) (Shah-Hosseini, 2007; Shah-Hosseini, 2009), Multiple Knapsack Problems (MKP) (Shah-Hosseini, 2008; Shah-Hosseini, 2009), n-queen puzzle (Shah-Hosseini, 2009), Air Robot Path Planning (Duan et al., 2008), single UCAV smooth trajectory

planning (Duan et al., 2009), Vehicle Routing Problem (Kamkar et al., 2010) and Economic Load Dispatch Problem (Rayapudi, 2011).

IWD algorithm is inspired from actions and reactions that happen between riverbed and swarm of water drops, in order to find the optimum or near optimum path from source of the water to its destination. IWD algorithm has some of the prominent characteristics of the water drops in a river and tries to imitate the actions that happen in a river between water drops and soil of the riverbed. In nature, the path in which a river flows has formed by a large swarm of water drops. The water drops change environment while they are flowing on the ground. On the other hand, the environment resists against movement of water drops such that parts having hard soil resist more than the parts having soft soil. The environment also may have different kinds of obstacles which may be considered as restrictions of the problem at hand. The gravitational force of the earth is the power source of water drops to move toward their destination which usually is a sea or lake. As a result of this movement, environment is changed during the time and the path to the destination emerges gradually.

Observation of the nature demonstrates that in spite of the obstacles of the environment and a lot of twists and turns, a river constructs its optimal or near optimal path to the destination. Similarly, IWD algorithm often tries to find the shortest path to the destination when the destination is known (Santos, 2009). In cases that the destination is unknown, the solution is obtained in terms of a quality measure or a kind of cost specified for the problem. In order to utilize this idea, an appropriate fitness function should be defined for the problem.

Artificial water drops try to emulate some main features of real water drops flowing in a river by making the following assumptions. The first one is that every water drop has its velocity in the river. The second assumption is that each water drop can carry an amount of soil which is skimmed from the riverbed (Shah-Hosseini, 2007). This assumption implies that soil of the riverbed can be transferred from one place to another, usually from the parts having strong currents to the parts having low currents. As a result of this process, fast parts get deeper and attract more water whereas removed soils are unloaded in slow parts of the riverbed making those

parts less attractive for the future water drops. Generally, when a water drop moves from one point to another, its speed and soil are increased and at the same time the soil of riverbed between the two points is decreased. The speed of water drop plays an important role in the amount of soil skimmed from the riverbed. Principles that are utilized in IWD algorithm may be summarized as follows regarding that Size of a water drop depicts its soil amount, length of an arrow shows the magnitude of the velocity, and direction of the arrow shows the direction of the movement.

Principle 1: Water drops flowing in a river carry an amount of soil which is removed from riverbed. Figure 3.8 illustrates that soil of a water drop increases while moving in the river and at the same time, soil of the riverbed is decreased.

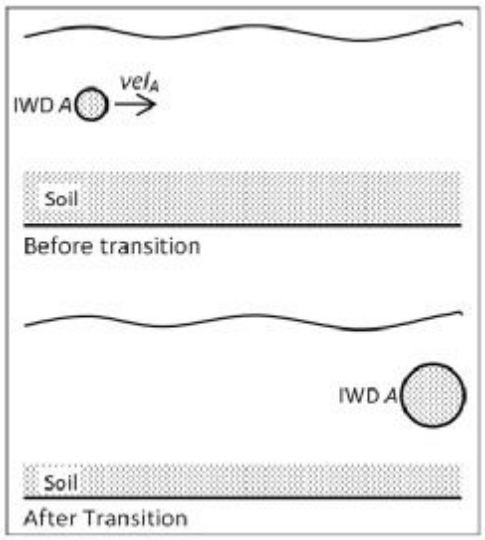


Figure 3.8. Illustration of Principle 1 for IWD algorithm.

Principle 2: It is mentioned that a water drop has also a velocity which determines the amount of the soil that is removed from the riverbed. Therefore, the quantity of soil which is removed by a water drop is proportional to its velocity. A faster water drop removes and gathers more soil than a slower one. In Figure 3.9, it is demonstrated that IWD *B* which is faster than IWD *A*, has gathered more soil when they reach to the right. Notice that soil amounts of the two water drops were equal at the beginning (left point).

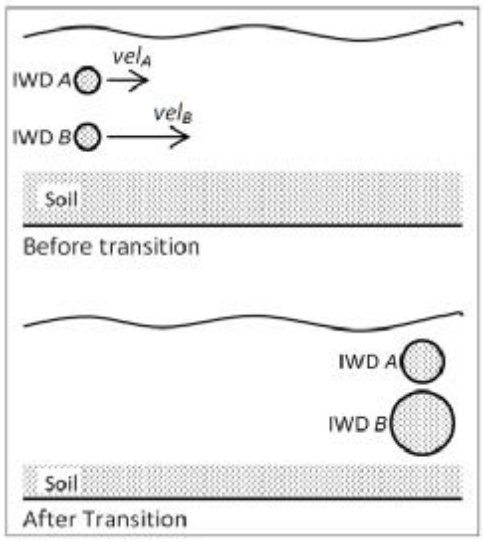


Figure 3.9. Illustration of Principle 2 for IWD algorithm.

Principle 3: A path with a lower amount of soil increases the velocity of a water drop more than a path having a higher amount of soil. Therefore, the soil amount of riverbed affects the amount of change in the velocity of water drops. In Figure 3.10, two similar water drops, IWD *A* and IWD *B* move on different paths, Path 1 and Path2 respectively. The water drop moving on path 2 gathers more soil and gains more speed than the water drop moving on path 1 which has a higher amount of soil than path 2. In fact, paths with higher amount of soils resist more against movement of water drops and let them gather less soil and gain less velocity.

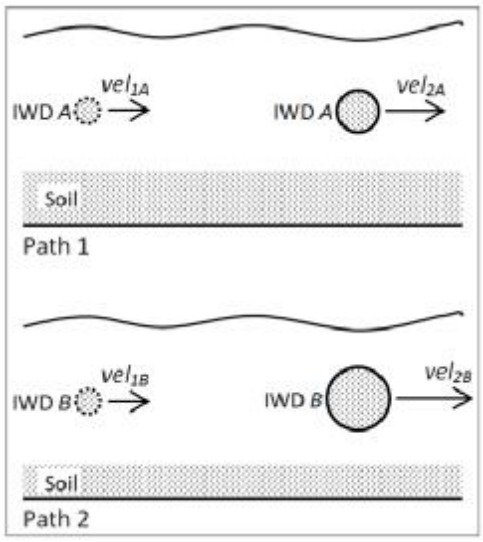


Figure 3.10. Illustration of Principle 3 for IWD algorithm.

Furthermore, a path selection strategy is also required regarding that there may be several branches from a point in the path to destination. Water drops prefer to select an easy path rather than a hard one. In IWD algorithm, the easiness or hardness of a path is determined by the amount of soil that exists on the path. Therefore, when water drops have to choose between several paths, they prefer paths with lower amounts of soil.

3.1.3.2. Intelligent Water Drops

This section provides an overview for prominent features of Intelligent Water Drops (IWDs) and introduces general forms of the related equations. In IWD algorithm, a problem is represented by a graph (N, E) which N is the set of nodes and E is the set of edges. This graph plays role of the environment for water drops, therefore edges of the graph are similar to paths where water drops flow in real world. In order to solve a problem, IWD algorithm employs a number of water drops, each one containing its own solution. This solution is gradually constructed for each water drop while travelling along the edges and visiting the nodes. After all water drops have completed their solution tours, the solutions are evaluated based on a fitness function.

It is assumed that IWDs move in a discrete environment and the length of paths or edges are finite. When an IWD moves from current node i to next node j, its velocity is increased by an amount of Δvel. As a more precise description of Principle 3, Δvel is nonlinearly proportional to the inverse of the soil that currently exists between points i and j, $soil(i, j)$. Nonlinear proportionality is denoted by $\propto^{NL}$ and general equation of Δvel is given by:

$$\Delta vel \propto^{NL} \frac{1}{soil(i,j)}$$

Based on Principle 1, soil of the IWD, $soil$(IWD), is also increased while moving from node i to j by removing some soil from the edge connecting two nodes. The amount that is added to the soil of the IWD, $\Delta soil$(IWD) is nonlinearly

proportional to the inverse of the time that is needed for IWD to flow from node i to j, denoted by $time(i, j)_{\text{IWD}}$. General equation of $\Delta soil$ is given by:

$$\Delta soil(\text{IWD}) \propto^{NL} \frac{1}{time(i,j)_{\text{IWD}}}$$

The time required to move from location i to location j is calculated by physics law of linear motion. Based on this law, the time is linearly proportional to the inverse of velocity. Regarding that each IWD has its own velocity, the required time for passing from node i to j will vary for different water drops. Linear proportionality is denoted by $\propto^{L}$ and general equation to calculate the transition time is given by:

$$time(i,j)_{\text{IWD}} \propto^{L} \frac{1}{vel(\text{IWD})}$$

Furthermore, the law of linear motion declares that the transition time is also proportional to the distance between two locations. Here, the effect of distance is represented by a local heuristic function $HUD(i, j)_{\text{IWD}}$ which stands for undesirability of an IWD to move from node i to node j. The heuristic function should be designed appropriately for a given problem. Thus, general equation for $time(i, j)_{\text{IWD}}$ is completed as follows:

$$time(i,j)_{\text{IWD}} = \frac{HUD(i,j)_{\text{IWD}}}{velocity(\text{IWD})}$$

The amount which is added to IWD's soil while moving from node i to j, $\Delta soil(\text{IWD})$, is used to update soil of the edge between two nodes. The updated amount, $soil(i, j)$, is linearly proportional to $\Delta soil(\text{IWD})$:

$$soil(i,j) \propto^{L} \Delta soil(\text{IWD})$$

Soil of the IWD is updated by:

$$soil_{t+1}(\text{IWD}) = soil_t(\text{IWD}) + \Delta soil(\text{IWD})$$

When IWDs have to choose a path among several paths, a probability function is utilized. The probability for an IWD to move from node i to node j, $p(i, j)_{\text{IWD}}$ is inversely proportional to the soil of edge between two nodes. Therefore, a path with a considerable amount of soil has less chance to be chosen by IWDs. The general equation is given by:

$$p(i,j)_{\text{IWD}} \propto^{L} \frac{1}{soil(i,j)}$$

Exact forms of the aforementioned equations are given in Section 3.2.1 which the IWD-CQ algorithm is described.

3.1.4. Differential Evolution Algorithm

3.1.4.1. Introduction

Differential Evolution (DE) algorithm is a fast and reliable global optimization method that is easy to implement and use. DE is based on the Genetic Annealing algorithm which was introduced by Price in 1994. Afterwards, Genetic Annealing algorithm was modified to be used with floating-point and arithmetic operations instead of bit-string encoding and logical operations respectively. The mentioned modifications along with a differential mutation operator provide the basis of transition from Genetic Annealing to DE algorithm (Storn, 1996; Storn and Price, 1997). Initially, the effectivenesss of DE algorithm was proved on Chebyshev polynomials. Today, DE has been applied to a wide range of scientific and engineering problems and it is known as an effective global optimizer. Although DE does not guarantee to find the optimal solution of the problem, it is able to explore large search spaces and find acceptable solutions in a reasonable amount of time. DE is a population based optimization algorithm which aims to find the optimal solution

by iteratively improving a population of candidate solutions according to a given measure of fitness quality.

Because DE algorithm does not need gradient of the problem, it can be used to solve non-differentiable optimization problems. In addition, the nature of the problem is not necessary to be known for optimization with DE algorithm. Therefore, only few or even no assumptions are needed to be made about the problem. These properties turn DE to an effective algorithm for continuous, non-continuous, noisy and dynamic problems.

3.1.4.2. Population

DE maintains two populations of candidate solutions; primary population, and temporary population. The reason to use separate primary and temporary populations is to provide better support for parallel machine architectures. Both of the populations contain N_P solution vectors which are composed of D real values ($N_P \geq 4$). Assuming that the primary population and the solution vectors are respectively denoted by $\mathrm{P}_{\mathbf{x}}$ and $\mathbf{x}_{i,g}$, structure of the population and solution vectors can be defined as follows (Price et al., 2005):

$$\mathrm{P}_{\mathbf{x},g} = (\mathbf{x}_{i,g}), \qquad i = 0,1,\dots,N_P - 1, \qquad g = 0,1,\dots,g_{\max},$$
$$\mathbf{x}_{i,g} = (x_{j,i,g}), \qquad j = 0,1,\dots,D - 1.$$

where N_P is the number of solution vectors inside each population, D is the number of parameters, and $g_{\max}$ is the maximum number of the generations that the algorithm is allowed to run. Index g specifies the generation which a solution vector belongs. The solution vectors are indexed with i inside the population, similarly, j indexes the parameters inside each solution vector. Primary population is initialized with random values from a uniform distribution.

In mutation stage, solution vectors are chosen randomly to be used in a mutation process. Result of the mutation process is a mutant solution vector which is inserted in an intermediary population. Mutation process is repeated until N_P mutant

solution vectors are produced and inserted in the intermediary population. Following terms define structures of the intermediary population $\mathrm{P}_{\mathbf{v},g}$ and the mutant vectors $\mathbf{v}_{i,g}$.

$$\mathrm{P}_{\mathbf{v},g} = (\mathbf{v}_{i,g}), \qquad i = 0,1,\dots,N_P - 1, \qquad g = 0,1,\dots,g_{\max},$$
$$\mathbf{v}_{i,g} = (v_{j,i,g}), \qquad j = 0,1,\dots,D - 1.$$

In recombination stage, each solution vector from the primary population is combined with a mutant solution vector from the intermediary population. The result of each recombination is a trial solution vector which overwrites the corresponding vector in the intermediary population. Therefore, there is no need to have separate intermediary and trial populations. The trial population and trial solution vectors are defined as follows.

$$\mathrm{P}_{\mathbf{u},g} = (\mathbf{u}_{i,g}), \qquad i = 0,1,\dots,N_P - 1, \qquad g = 0,1,\dots,g_{\max},$$
$$\mathbf{u}_{i,g} = (u_{j,i,g}), \qquad j = 0,1,\dots,D - 1.$$

DE algorithm is composed of four major steps illustrated in Figure 3.11; Initialization, Mutation, Recombination, and Selection.

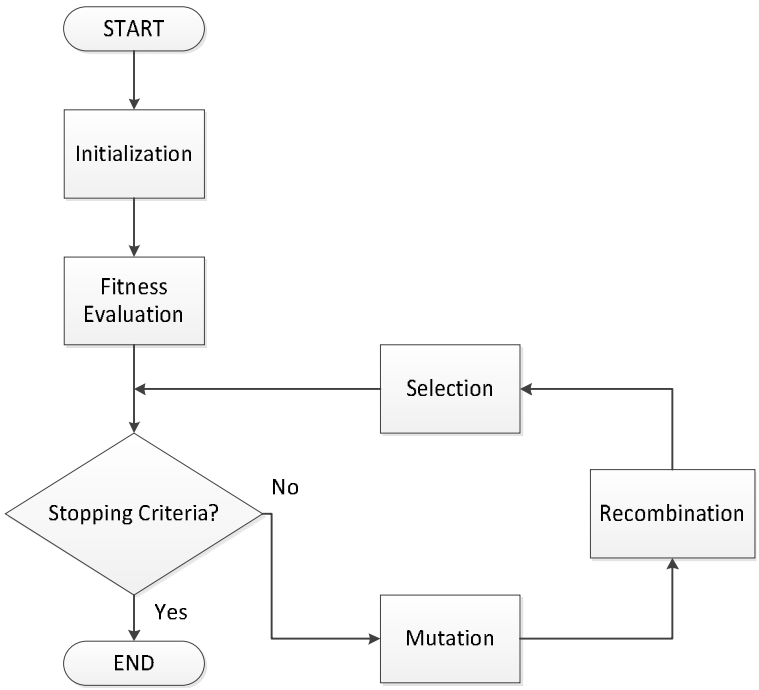


Figure 3.11. Flowchart of standard DE algorithm.

3.1.4.3. Initialization

Previously, it was mentioned that at the beginning of the algorithm the primary population is initialized with uniformly distributed random values. However, before the initialization, lower and upper bounds must be defined for each parameter. Vectors $\mathbf{b}_{\mathrm{L}}$ and $\mathbf{b}_{\mathrm{U}}$ specify the lower bounds and the upper bounds of the parameters respectively. The random values for each parameter are generated within the correspondent range specified by these vectors. The initial value (g=0) of j^{th} parameter of the i^{th} solution vector is specified as follows:

$$x_{j,i,0} = rand_j(0,1) \cdot \left(b_{j,\mathrm{U}} - b_{j,\mathrm{L}}\right) + b_{j,\mathrm{L}}, \qquad 0 \leq rand_j(0,1) < 1$$

Subscript j in $rand_j(0,1)$ indicates that for each parameter, a new random value is generated.

3.1.4.4. Mutation

In mutation stage an intermediary population is produced which contains N_P mutant solution vectors. Various differential mutation operators can be used to obtain the mutant solution vectors (Storn, 1996). The standard mutation operator of DE algorithm to generate mutant vectors is given in the following equation:

$$\mathbf{v}_{i,g} = \mathbf{x}_{r0,g} + F \cdot (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}) \qquad (3.1.)$$

where F denotes the differential weight which is a user-defined real value from the range [0,2] (Pedersen and Chipperfield, 2008). This value controls the rate at which the population evolves. For producing each mutant vector, three vectors; $r0$, $r1$ and $r2$ are chosen randomly from the primary population. These vectors must be different from each other and from the target vector, $\mathbf{x}_{i,g}$. An example of obtaining a mutant solution vector in a two-dimensional parameter space is demonstrated in Figure 3.12.

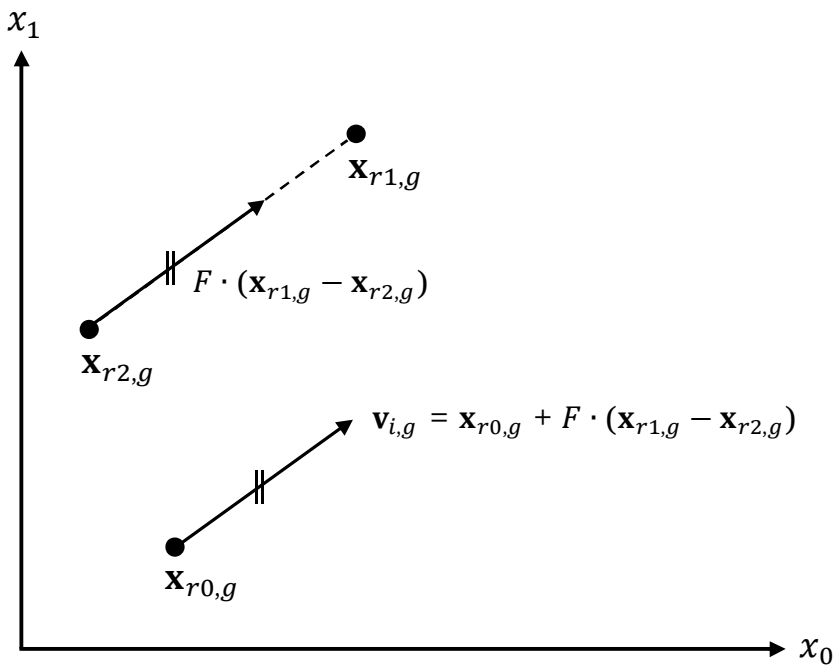


Figure 3.12. Differential mutation: mutant vector $\mathbf{v}_{i,g}$ is produced by adding weighted differential, $F \cdot (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g})$, to the base vector $\mathbf{x}_{r0,g}$. The value of differential weight, *F*, is approximately 0.7 here.

3.1.4.5. Recombination

A uniform crossover recombination method is utilized in DE algorithm to enhance its searching capabilities. Each solution vector $\mathbf{x}_{i,g}$ from the primary population is crossed with a mutant vector $\mathbf{v}_{i,g}$ from the intermediary population. The result is a trial vector $\mathbf{u}_{i,g}$:

$$\mathbf{u}_{i,g} = u_{j,i,g} = \begin{cases} v_{j,i,g} & \text{if}\left(\text{rand}_j(0,1) \leq CR \text{ or } j = j_{\text{rand}}\right) \\ x_{j,i,g} & \text{otherwise} \end{cases} \qquad (3.2.)$$

CR, Crossover probability, is a user-defined real value from the range [0,1]. *CR* controls the fraction of parameters that are selected from the mutant vector. For each parameter, a random number, $\text{rand}_j(0,1)$, is generated. If the generated number is less than or equal to *CR*, the parameter is copied from the mutant vector $\mathbf{v}_{i,g}$, otherwise, it is taken from the solution vector $\mathbf{x}_{i,g}$. Based on the value of *CR*, it is possible that no parameter be selected from the mutant vector. In this case, the trial

vector, $\mathbf{u}_{i,g}$, will be a duplicate of the target vector $\mathbf{x}_{i,g}$. In order to prevent this problem, it should be ensured that at least one parameter will be taken from the mutant vector. In this regard, a random index, $j_{rand} \in [1..D]$, is generated. If parameter index j is equal to j_{rand}, the correspondent parameter is always copied from the mutant vector regardless of CR.

After calculation of each trial vector, it overwrites the correspondent mutant vector in the intermediary population. As recombination process progresses, intermediary population is gradually replaced with trial vectors population.

3.1.4.6. Selection

Selection process determines the next generation of the DE algorithm. The selection is based on the fitness values that are obtained from fitness function $f(.)$. If the fitness value of target vector $\mathbf{x}_{i,g}$, is higher than the fitness value of trial vector, $\mathbf{u}_{i,g}$, it is replaced by the trial vector. Otherwise, the target vector is retained in the next generation.

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g} & \text{if } f(\mathbf{u}_{i,g}) \leq f(\mathbf{x}_{i,g}) \\ \mathbf{x}_{i,g} & \text{otherwise} \end{cases}$$

It can be seen that the existing primary population is gradually overwritten by the next population. The same process takes place when the intermediary population is replaced by the trial population. Therefore, two populations are enough to be held at each generation of the algorithm.

3.1.4.7. Termination

After installation of the next population, three major steps; mutation, recombination, and selection are repeated until a stopping criterion is met. Possible stopping criteria can be finding the desired optimum solution or reaching the maximum calculation budget (for instance, the maximum number of generations, g_{max}).

3.2. Method

3.2.1. CQ with Intelligent Water Drops Algorithm (IWD-CQ)

3.2.1.1. Overview

In this section, implementation details of IWD algorithm to Color Quantization are described. The first step in IWD based algorithms is to define an appropriate environment for the IWDs. Here, it is assumed that images are given in gray-scale. Pixels of a gray-scale image may have 256 different gray colors from 0 to 255. If each color is considered as a node, there will be 256 nodes to move on. These nodes are fully connected. Therefore, the environment of the problem is specified by a 256×256 matrix and is denoted by $soil(i, j)$. Each element of the *soil* matrix with arbitrary coordinates of i and j, represents the amount of soil that currently exists on the path between nodes i and j. Self-connection is not allowed for the nodes. Thus, elements of the matrix placed on main diagonal remain zero during the execution of the algorithm. Other elements will be symmetric about the main diagonal because the path connecting node i to j is conceptually same as the path that connecting node j to i. More specifically:

$$soil(i,i) = 0 \quad \text{and} \quad soil(i,j) = soil(j,i) \quad \forall\, i,j \in [0..255]$$

Each IWD is initially placed on a node which is considered as its first visited node in the solution tour T^{IWD}. Then it moves to another unvisited node until the required number of nodes is reached that is equal to the desired palette size. Therefore, the solution tour of each IWD corresponds to a candidate palette for quantization process.

3.2.1.2. IWD-CQ Algorithm

Following steps are used for implementation of the proposed algorithm, IWD-CQ. In addition, four heuristics are introduced and explained in Step 2.

Step 1: Initialize the following static parameters:

iterMax is the maximum number of iterations.

reinitNo is used to trigger the reinitialization process which helps the algorithm to evade from possible local optima. *reinitNo* is a positive integer and was determined empirically as 30.

initSoil is the initial amount of soil on all paths. It is a user defined positive value. *initSoil* is selected to be 1000 as suggested in (Shah-Hosseini, 2007). The *soil* matrix is initialized as follows:

$$soil(i,j) = initSoil \qquad \forall\, i,j \in [0..255]\ and\ i \neq j$$

nColor is the number of colors in desired palette.

nIWD is the total number of water drops employed in the algorithm and is calculated by;

$$nIWD = nIWDperNode \times trimmedGrayLevels$$

where, *nIWDperNode* is the number of water drops allocated for each node and *trimmedGrayLevels* is the range of gray colors used in the original image. *nIWDperNode* is a user selected positive integer and in this thesis it is set to 1. *trimmedGrayLevels* is an image dependent value which helps to increase the efficiency of the algorithm by preventing the allocation of IWDs to the gray levels which fall outside of the color range.

initVel is the initial velocity of IWDs. *initVel* is a user selected positive value. Here, it is specified as 100 based on (Shah-Hosseini, 2007).

a_v, b_v and c_v are velocity updating parameters. In this thesis a_v=1000, b_v=0.01 and c_v=0.1 are used.

a_s, b_s and c_s are soil updating parameters. In this thesis a_s=1000, b_s=0.01 and c_s=0.1 are used.

Step 2: Initialize IWDs by setting; soil amount *soil*(IWD) to zero, velocity *vel*(IWD) to *initVel* and visited node list *vn*(IWD) to an empty list. This list will represent the solution tour, T^{IWD} obtained by the IWD. Furthermore, the heuristic

function of each IWD, $HUD(j)_{IWD}$, is initialized. $HUD(j)_{IWD}$ provides undesirability level of the node j for IWD. Four different heuristic functions are suggested in this thesis. These are:

a. Con-HUD: In this method, a constant undesirability value is assigned to every node j in HUD_{IWD}. The value is specified empirically as 300. Therefore, HUD_{IWD} is initialized as follows:

$$HUD(j)_{IWD} = 300 \quad \forall j \in [0..255]$$

b. Pop-HUD: In this method, inverse of the histogram is used as the heuristic. Histogram of an image demonstrates frequencies of the colors used in the image. Therefore, when inverse of the histogram is used as heuristic, it provides higher undesirability values for less frequent colors and lower undesirability values for more frequent colors. As the result, IWDs tend to select more popular colors while less popular colors have still chance to be chosen. Assuming that the frequency of color j is denoted by $hst(j)$, the heuristic function is given by:

$$HUD(j)_{IWD} = \alpha \cdot \left(\frac{\left((hst_{\max} - hst_{\min}) - (hst(j) - hst_{\min}) \right)}{(hst_{\max} - hst_{\min})} \right) + \beta \qquad (3.3.)$$

$$\forall j \in [0..255]$$

where, α and β are used to scale and shift the heuristic values, and hst_{min} and hst_{max} are respectively the minimum and the maximum frequency of the colors in histogram. Here, α and β are chosen to be 200 and 100 respectively.

c. Med-HUD: In this method a virtual dynamic histogram, $dynHst_{IWD}$ is used to calculate the heuristic function. Thus, HUD_{IWD} is not required to be initialized here while $dynHst_{IWD}$ is initialized as follows:

$$dynHst(j)_{IWD} = 1 \quad \forall j \in [0..255]$$

d. P-M-HUD: This method is similar to Method c with a different initialization of the $dynHst_{IWD}$ as follows:

$$dynHst(j)_{\text{IWD}} = hst(j) \quad \forall j \in [0..255]$$

Step 3: Distribute IWDs uniformly over the nodes. As mentioned earlier, *nIWDperNode* determines the number of IWDs allocated for each node. When IWDs are being distributed, the correspondent node is inserted in list *vn*(IWD).

Step 4: Skip if Method c or d is not selected in Step 2. Divide color range into sub-ranges by using visited node list, *vn*(IWD). Amplify the longest sub-range of $dynHst_{\text{IWD}}$ by 20 percent, and subside other sub-ranges by 20 percent. This makes the longest color range more appealing for the next selections. This behavior is inspired from the median-cut CQ algorithm. In this case, heuristic function HUD_{IWD} is calculated by substituting *hst* with $dynHst_{\text{IWD}}$ in Equation 3.3.

Step 5: Make undesirable each visited node and its close vicinity in *vn*(IWD). This significantly reduces the probability of a node to be chosen in the neighborhood of a previously selected node. Unless Method a is selected in Step 2, *HUD* values of node *j* and its neighborhood nodes are extremely amplified as follows:

$$HUD(n)_{\text{IWD}} = HUD(n)_{\text{IWD}} + \theta \cdot \left(e^{\frac{-(n-j)^2}{2\sigma^2}} \right)$$
$$\forall n \in [j-5..j+5] \quad and \quad \forall j \in vn(\text{IWD})$$

where σ denotes spread of the Gaussian distribution and θ is a fairly large number used as coefficient for the Gaussian distribution. In this thesis, σ =0.5 and θ=2000.

Step 6: For each IWD currently placed on node *i*, calculate the probability of moving to every next node *j* by:

$$p(i,j)_{\text{IWD}} = \frac{f\big(soil(i,j)\big)}{\sum_{k \notin vn(\text{IWD})} f\big(soil(i,k)\big)}$$

such that

$$f\big(soil(i,j)\big) = \frac{1}{\varepsilon_s + g\big(soil(i,j)\big)}$$

and

$$g\big(soil(i,j)\big) = \begin{cases} soil(i,j)\ if \ \min\limits_{l \notin vn(\mathrm{IWD})}\big(soil(i,l)\big) \geq 0 \\ soil(i,j) - \min\limits_{l \notin vn(\mathrm{IWD})}\big(soil(i,l)\big)\ else \end{cases}$$

where ε_s is an small positive value ($\varepsilon_s = 0.01$) to prevent a possible division by zero in function $f(.)$. Function $g(.)$ is used to shift the given soil amount to a higher value when minimum of the soil amounts of paths from current node to unvisited nodes is negative. After calculating the probabilities, roulette wheel selection method is used to identify the next node j. Then IWD moves to the selected node and the node is added to visited nodes list vn(IWD).

Step 7: Update the velocity of IWD as in (Shah-Hosseini, 2008) by using following equation:

$$vel_{t+1}(\mathrm{IWD}) = vel_t(\mathrm{IWD}) + \frac{a_v}{b_v + c_v \cdot soil^2(i,j)}$$

where vel_{t+1}(IWD) denotes the updated velocity of the IWD and $soil^2(i, j)$ is square of the soil amount of the path joining two nodes i and j.

Step 8: Calculate the amount of soil which will be added to IWD's soil as follows:

$$\Delta soil(\mathrm{IWD}) = \frac{a_s}{b_s + c_s \cdot time^2(i,j)_{\mathrm{IWD}}}$$

such that

$$time(i,j)_{\mathrm{IWD}} = \frac{HUD(j)_{\mathrm{IWD}}}{vel(\mathrm{IWD})}$$

Step 9: Update soil of the IWD, $soil$(IWD) and soil of the path joining node i to node j, $soil(i, j)$ as follows:

$$soil_{t+1}(\mathrm{IWD}) = soil_t(\mathrm{IWD}) + \Delta soil(\mathrm{IWD})$$

and

$$soil(i,j) = (1-\rho) \cdot soil(i,j) - \rho \cdot \Delta soil(\mathrm{IWD})$$

where parameter ρ is a positive value less than one. In this thesis ρ=0.1 is used. Repeat steps 4-9 until all IWDs complete their solution tours.

Step 10: Evaluate the fitness of solution tours of all IWDs by using Mean Absolute Error (MAE). MAE of a solution tour can be calculated by using Equation 3.4.

$$MAE = \frac{1}{M}\sum_{i=1}^{M}|I_i - Q_i| \tag{3.4.}$$

where I_i and Q_i are the values of i^{th} pixel in original and quantized images and M is the total number of pixels. If we denote the fitness of a solution tour s with $qnte(s)$, iteration best solution tour, T^{IB} is determined as follows:

$$T^{IB} = \arg\left(\min_{\text{for all IWDs}} qnte(T^{\text{IWD}})\right)$$

where T^{IWD} denotes the solution tour obtained by each IWD and arg(.) returns its argument which is the solution tour that has produced the minimum quantization error.

Step 11: Update the paths which are included in iteration best solution tour as follows:

$$soil(i,j) = (1-\rho)\cdot soil(i,j) - \rho\cdot\left(\frac{soil(IWD)_{IB}}{nColor-1}\right)$$
$$\forall(i,j) \in T^{IB}$$

where $soil(\text{IWD})_{IB}$ is the soil amount of IWD possessing the iteration best solution tour.

Step 12: If iteration best solution is better than the best solution found so far (T^{TB}), then update total best tour T^{TB} as:

$$T^{TB} = T^{IB} \quad if \quad qnte(T^{IB}) < qnte(T^{TB})$$

Step 13: Launch the reinitialization process if the algorithm is repeated *reinitNo* times. During the process, soils of all paths are set to *initSoil* except the

paths included in total best solution. In order to retain the attractiveness of the paths included in total best solution, their soils are set to α percent of the *intiSoil*. In this study α is chosen as 20. Repeat Steps 2-13 until the maximum number of iterations is reached or until any other terminating criterion is met.

At the end of the algorithm, T^{TB} represents the best solution tour which is found by IWDs during the execution of the algorithm. T^{TB} is used as desired palette for final quantization process.

3.2.2. Perceptual CQ with Differential Evolution Algorithm (DE-PCQ)

3.2.2.1. Overview

Regarding that the final result of the color quantization process is observed and evaluated by the human, CQ methods should not only reduce the quantization error but also consider the perception of human vision. Perceptual CQ methods aim to obtain better quantization results by taking into account various features of the Human Visual System (HVS). Literature review presented in Section 2.3 shows that different models and approaches have been employed in this regard.

In this section, a Perceptual CQ method is investigated using Differential Evolution algorithm as the optimization tool. The proposed method is based on a particular feature of the HVS which is related to the perceived color of adjacent pixels. It is known that human's eye perceives an average of the colors of adjacent pixels when the pixels are relatively far from the eye (Puzicha et al., 2000). Here, the mentioned property is utilized to propose a perceptual CQ approach with DE algorithm.

3.2.2.2. The Perceptual Approach

Human Visual System possesses numerous psychophysical characteristics that are used as the inspiration source for perceptual color quantization models. Digital halftoning technics make additional perceived colors based on the fact that

the human eye perceives a spatial averaging of the adjacent colors (Kandel et al., 1991). Thus, this method can virtually produce new colors which do not actually exist (Kolpatzik and Bouman, 1992). The proposed algorithm utilizes this property by using a halftoning technique which propagates the quantization error of each pixel to its neighboring pixels.

In addition, the mentioned property of HVS is simulated directly in the algorithm. This lets the algorithm to minimize the quantization error according to the final perception of HVS from the quantized images. The averaging characteristic of human eye is modeled using a Gaussian filter which applies a slight smoothing on the images. In order to present an empirical illustration, two identical gray-scale images are shown in Figure 3.13. The left image is a halftoned gray-scale image. The right image represents the same image after applying a slight smoothing. In a close observation of the images it can be seen that the left image seems ragged where the right one seems smooth. When the images are observed from a relatively far distance (approximately one meter) averaging property of the eye applies and consequently, both of the images seem exactly the same.

a.  b. 

Figure 3.13. Effect of smoothing filter; a) A halftoned gray-scale image; b) The image after smoothing.

In two dimensions, the Gaussian filter is defined by the following equation (Haddad and Akansu, 1991).

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad (3.5.)$$

where x and y represent the distances from the center in the horizontal and vertical axes respectively and σ is the standard deviation of the Gaussian distribution.

Figure 3.14 demonstrates the shape of the Gaussian filter in two dimensions. Values from this distribution are used to build a convolution matrix which is applied to the original image. Each pixel's new value is set to a weighted average of that pixel's neighborhood. This results in a blurring effect which is controlled by the σ value. The bigger value for σ produces more blurriness. Here, only a slight blurring effect is applied in order to simulate the averaging property of the HVS.

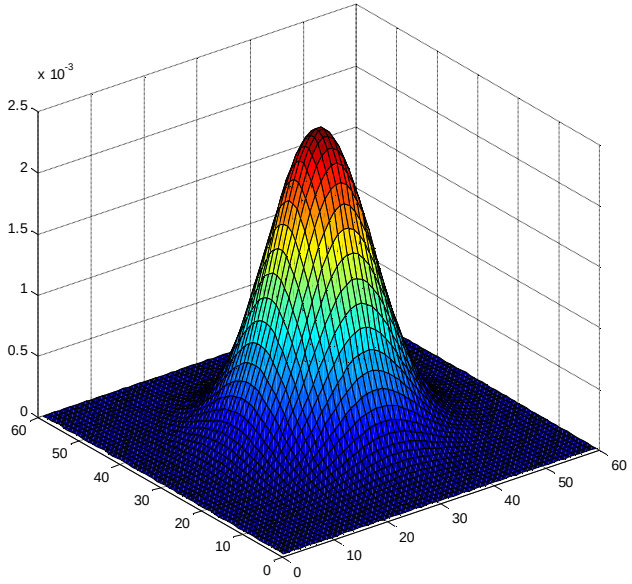


Figure 3.14. Shape of a typical Gaussian function in two dimensions.

Since the aim of study is to investigate a perceptual approach for color quantization, the difference between colors should be measured in a psychophysically meaningful measure of color similarity. In other words, Euclidian distance between colors should be perceived uniform to HVS. This prerequisite can be achieved by using a uniform color space (Sharma and Trussell, 1997). The RGB color space which is commonly used in Image Processing applications does not provide this feature. However, in CIE $L^*a^*b^*$ and CIE $L^*u^*v^*$ color spaces (L'Eclairage, 1986) Euclidian distances between colors are presented perceptually uniform to HVS.

In this section, CIE $L^*a^*b^*$ (CIELab) color space is implemented in the proposed algorithm. Detailed Explanations of the RGB and CIELab color spaces are presented in Section 3.1.2. Moreover, ΔE error measure (Pedersen and Hardeberg,

2009) is used to calculate the quantization error between original and quantized images. Assuming that x and y represent the i^{th} pixel in original and quantized images respectively, Euclidian distance between their colors are calculated as follows.

$$\Delta E_i = \sqrt{(L_x - L_y)^2 + (a_x - a_y)^2 + (b_x - b_y)^2}$$

For an image containing M pixels, quantization error, ΔE, is obtained by:

$$\Delta E = \frac{1}{M}\sum_{i=1}^{M} \Delta E_i \qquad (3.6.)$$

In order to observe the effect of implemented perceptual method, two different objective functions are investigated; objective function for basic CQ (*objFuncBCQ*), and objective function for perceptual CQ (*objFuncPCQ*). The *objFuncBCQ* is used to minimize the quantization error without using any halftoning method or blurring effect. It provides the basic color quantization for the optimization process. In contrast, *objFuncPCQ* uses error diffusion halftoning technique and applies Gaussian smoothing filter as described previously. The flowchart of objective functions is shown in Figure 3.15.

The flowchart of the objective function for basic CQ which is illustrated in Figure 3.15a starts with inputting color palette and the original image. The color palette is generated and passed by the optimization algorithm, here, Differential Evolution. Then, basic color quantization process is performed using the input palette. Quantization error between the original and quantized images is calculated in the next step. Since CIELab color space is used in the algorithm, the quantization error is calculated in terms of ΔE defined by Equation 3.6. Final step returns the value of ΔE as the result of process to DE algorithm.

The flowchart of the objective function for Perceptual CQ which is illustrated in Figure 3.15b repeats almost the same steps with a different CQ approach. Here, the quantized image is obtained employing Stucki error diffusion method as a

Halftoning technique. Then, the image is slightly blurred using a Gaussian smoothing filter which models the aforementioned property of HVS.

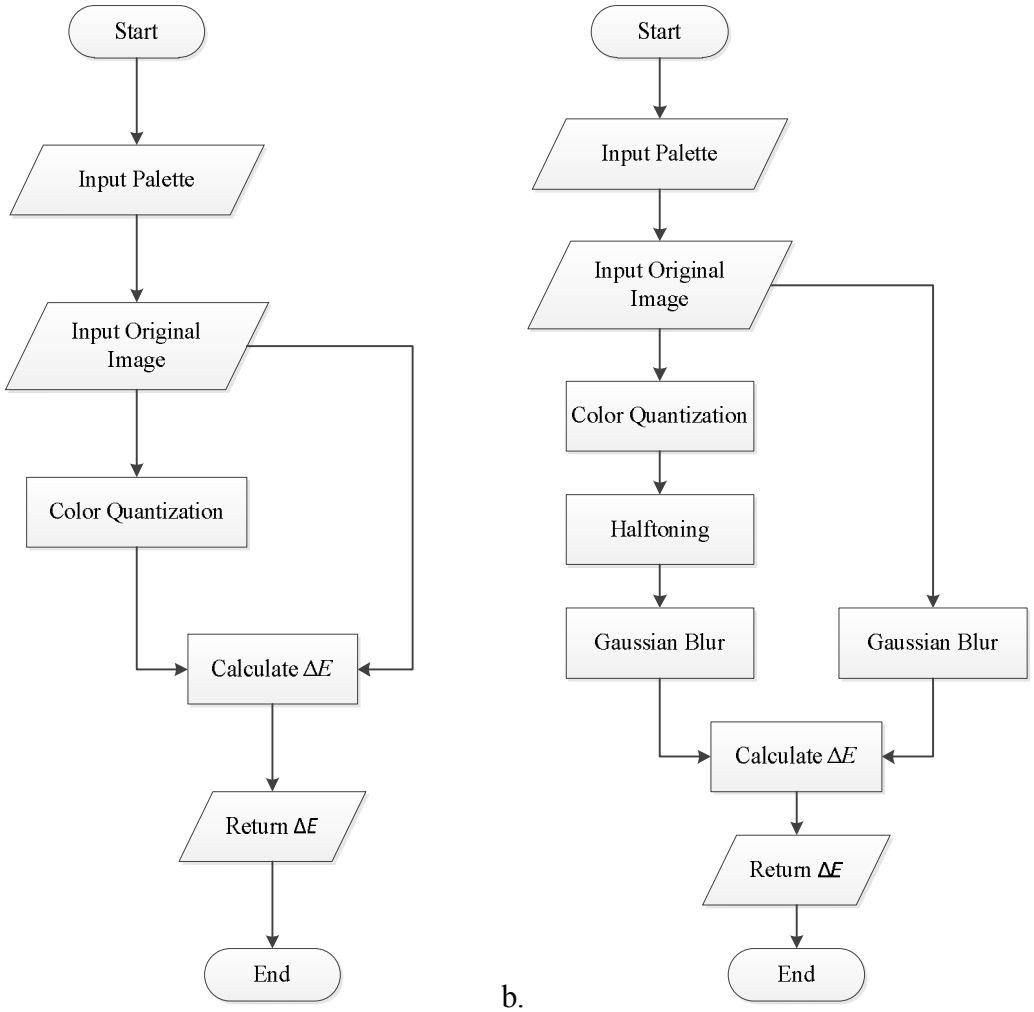


Figure 3.15. Flowchart of objective functions of DE-PCQ algorithm.
a) Objective function (*objFunc*) for basic CQ, *objFuncBCQ*.
b) Objective function for perceptual CQ, *objFuncPCQ*.

The explained objective functions are employed by DE algorithm in order to perform basic and perceptual color quantization processes. The flowchart of DE algorithm is presented in Figure 3.16. Previously, it was mentioned that the standard version of DE algorithm is implemented in the proposed approach. The flowchart given in Figure 3.16 is an adaptation of the standard DE algorithm (Figure 3.11) to CQ problem. The steps in Figure 3.16 are discussed in details in Section 3.2.2.3.

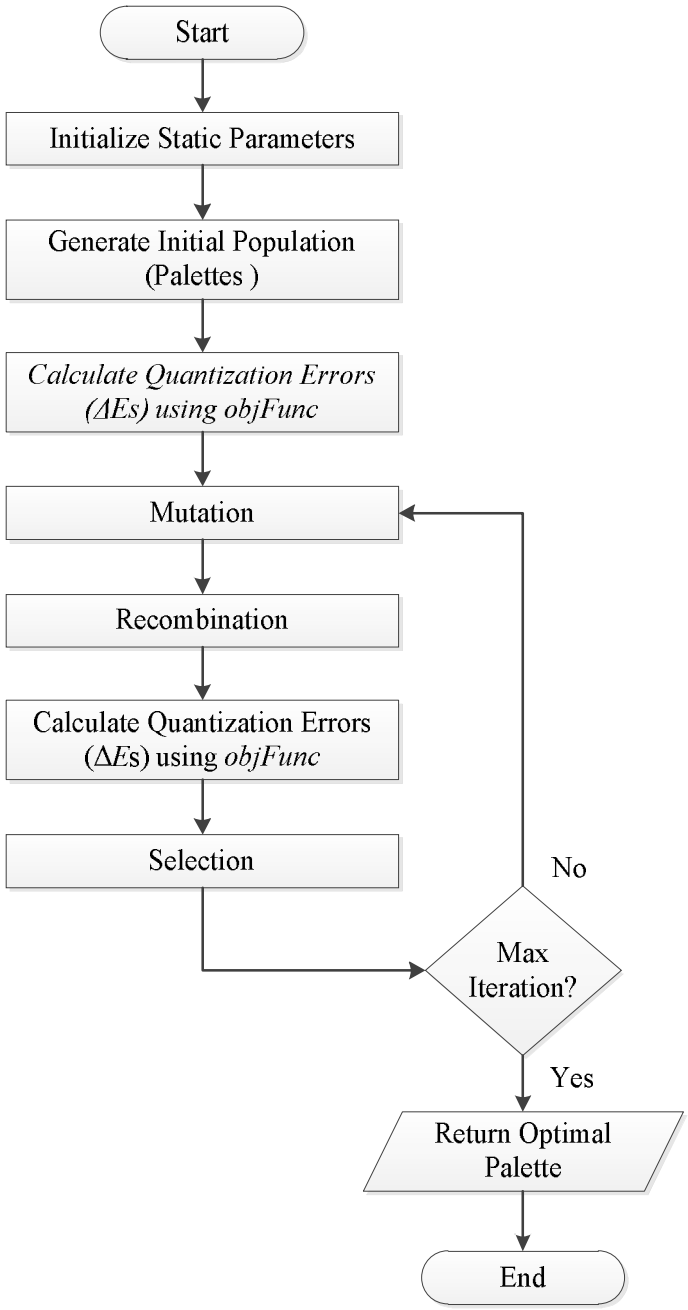


Figure 3.16. Sub-algorithm for acquiring the optimal palette with DE.

DE algorithm which is given in Figure 3.16 is utilized as a sub-process by the main approach, DE-PCQ, which is shown in Figure 3.17. It should be noted that after acquiring the optimal palette by DE algorithm, it is used to produce two quantized images; one with Halftoning and the other without Halftoning. Both of the quantized images are passed from a Gaussian filter and then their quantization errors are calculated according to the original image which is also passed from the same

Gaussian filter. The filter simulates the perceptual model of HVS for the algorithm. More details about the steps given in Figure 3.17 can be found in Section 3.2.2.3.

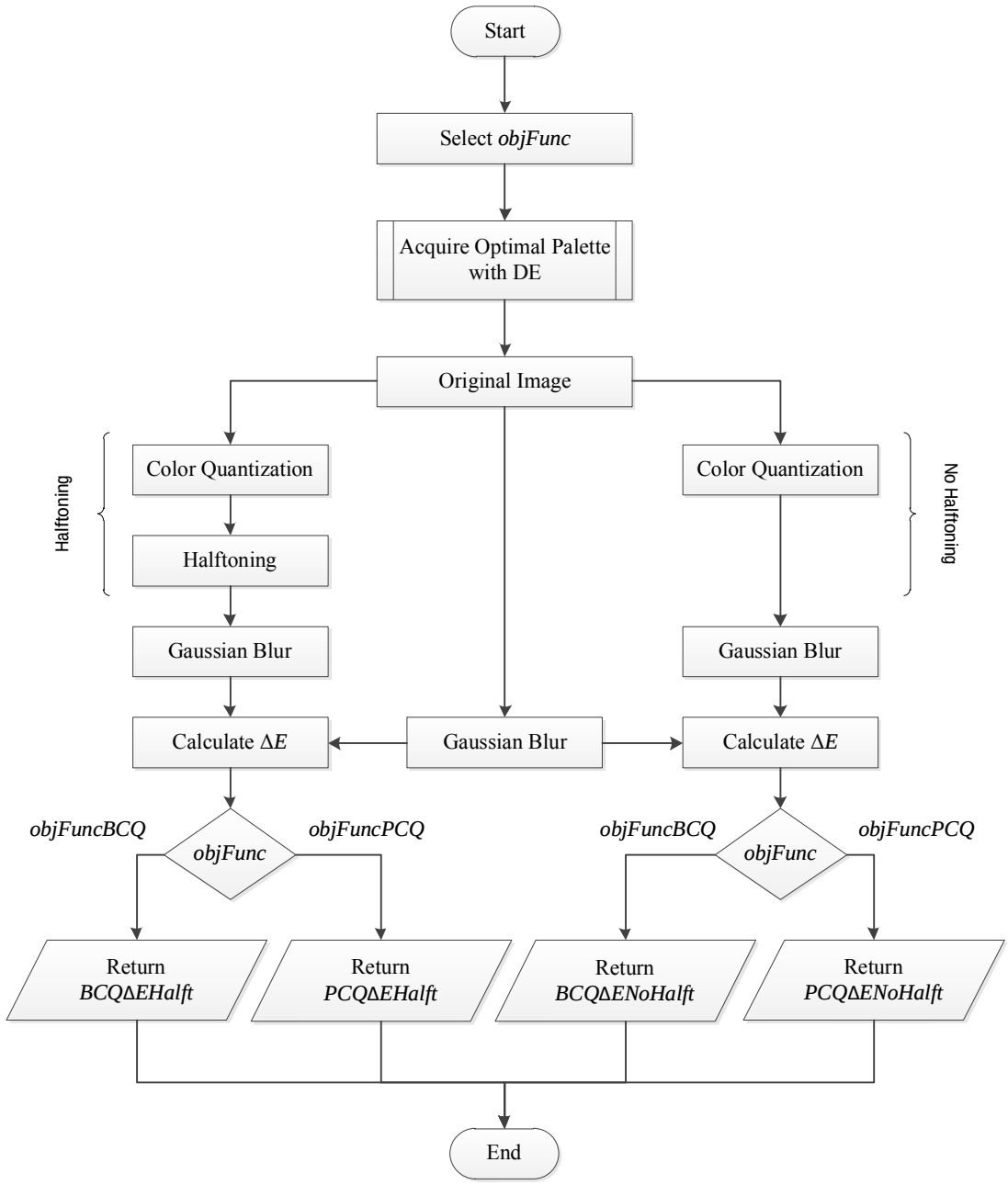


Figure 3.17. Main flowchart DE-PCQ.

3.2.2.3. DE-PCQ Algorithm

Implementation details of the proposed algorithm, DE-PCQ, are described in the following steps.

Step 1: Initialize the following static parameters:

g_{max} is the maximum number of generations that the algorithm is allowed to run. Here, gmax=1000.

nColor is the size of the desired color palette. Here, *nColor*=8.

F is the differential weight which should be chosen properly from the range [0,2]. The proper value is investigated in Section 4.

CR is the crossover rate which should be chosen properly from the range [0,1]. The proper value is investigated in Section 4.

N_P specifies the size of primary and temporary populations. The value of N_P is determined as 240 based on the following equation (Storn, 1996):

$$N_p = 10 \times D \tag{3.7.}$$

where $D = 3 \times nColor$ is the number of parameters to be tuned. It should be noted that the color of each pixel in CIELab color space is specified by three components; L, a, and b.

σ is the standard deviation of the Gaussian distribution for the Gaussian filter. Here, σ is determined empirically as 0.65.

objFunc which is also denoted by *f*(.), determines the objective function that is used during the optimization process. Objective functions are; *objFuncBCQ*, and *objFuncPCQ*.

Step 2: Construct the first generation using N_P solution vectors. Each vector is composed of D randomly generated values in the format; L_1, a_1, b_1, L_2, a_2, b_2, …, L_{nColor}, a_{nColor}, b_{nColor} and within the following ranges.

$$L \in [0,100], \quad a \in [-127,128], \quad b \in [-127,128]$$

Step 3: Evaluate the solution vectors and determine their fitness values (ΔEs) using the chosen objective function $f(.)$. Then, save the best vector as $\mathbf{x}_{tbest}$.

Step 5: Mutation: For each solution vector $\mathbf{x}_i$, select three solution vectors $\mathbf{x}_{r0}$, $\mathbf{x}_{r1}$, and $\mathbf{x}_{r2}$ randomly. The selected vectors must be different from each other and from the vector $\mathbf{x}_i$. Generate the correspondent mutant vector $\mathbf{v}_i$ using the following equation and add it in the intermediary population.

$$\mathbf{v}_i = \mathbf{x}_{r0} + F \cdot (\mathbf{x}_{r1} - \mathbf{x}_{r2}), \qquad i \in [1..N_P]$$

Step 6: Recombination: for each solution vector $\mathbf{x}_i$ from the primary population select its correspondent mutant vector $\mathbf{v}_i$ from the intermediary population and generate a trial vector $\mathbf{u}_i$ by applying uniform crossover according to the following equation,

$$\mathbf{u}_i = u_{j,i} = \begin{cases} v_{j,i} & \text{if}\left(\text{rand}_j(0,1) \le CR \text{ or } j = j_{\text{rand}}\right) \\ x_{j,i} & \text{otherwise} \end{cases}$$

where $i \in [1..N_P]$ indexes the vectors in the population and $j \in [1..D]$ indexes the parameters in the solution vector. j_{rand} is a random index from the range [1..D] which is regenerated per each solution vector. $rand_j(0,1)$ is a random real number from the range [0,1] that is regenerated per each parameter. Replace each mutant vector $\mathbf{v}_i$ with the correspondent trial vector $\mathbf{u}_i$. As the result, intermediary population alters to the trial population.

Step 7: Selection: Evaluate solution vectors in the trial population and determine their fitness values using the objective function $f(.)$. If a trial vector has better fitness value than its correspondent vector in the primary population, replace the primary solution vector with the trial one.

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g} & \text{if } f(\mathbf{u}_{i,g}) \le f(\mathbf{x}_{i,g}) \\ \mathbf{x}_{i,g} & \text{otherwise} \end{cases}$$

Step 8: Find generation best solution vector, $\mathbf{x}_{gbest}$, and update the total best solution vector $\mathbf{x}_{tbest}$ as follows:

$$\mathbf{x}_{\text{gbest}} = \arg\left(\min_{\forall i \in [1..N_P]} \Delta E(\mathbf{x}_i)\right)$$

$$\mathbf{x}_{\text{tbest}} = \mathbf{x}_{\text{gbest}} \quad if \quad \Delta E(\mathbf{x}_{\text{gbest}}) < \Delta E(\mathbf{x}_{\text{tbest}})$$

If the maximum number of generations, $g_{\max}$, is not reached, go to Step 5. Otherwise, continue with Step 9.

Step 9: Quantize the input image using $\mathbf{x}_{\text{tbest}}$ as color palette and obtain *qntImgNoHalft* which is the quantized image without halftoning.

Step 10: Quantize the input image using $\mathbf{x}_{\text{tbest}}$ as color palette and obtain *qntImgHalft* which is the quantized image with halftoning.

Step 11: Apply the Gaussian filter $G(x,y)$ defined by Equation 3.5 on the input image and quantized images; *qntImgNoHalft* and *qntImgHalft*.

Step 12: Calculate quantization error, ΔE, for the quantized images using Equation 3.6. In case *objFunc* = *objFuncBCQ*, the quantization errors for *qntImgNoHalft* and *qntImgHalft* are named *BCQΔENoHalft and BCQΔEHalft* respectively. If *objFunc* = *objFuncPCQ*, the quantization errors for *qntImgNoHalft* and *qntImgHalft* are named *PCQΔENoHalft and PCQΔEHalft* respectively.

4. RESULTS AND DISCUSSIONS

4.1. IWD-CQ

A comparative work has been carried out in order to evaluate the performance of the proposed IWD-CQ algorithm in terms of visual quality of the quantized image. The comparisons were made against eight algorithms which were organized in two different groups; conventional and artificial intelligence based (AI) algorithms. Conventional color quantization algorithms; Uniform CQ (U-CQ), Popularity CQ (P-CQ) (Boyle and Lippman, 1978), Median-Cut CQ (MC-CQ) (Heckbert, 1980), Octree CQ (O-CQ) (Gervautz and Purgathofer, 1988) and K-means CQ (Km-CQ) (Lloyd, 1957) are selected for benchmarking since they are well-known and widely used algorithms in the literature.

AI algorithms that are chosen for CQ benchmarking consist of; Kohonen's Self Organizing Maps (SOM), Ant Colony Optimization (ACO) and Simulated Annealing (SA). Regarding that IWD-CQ uses a heuristic, ACO was selected as the most similar heuristic algorithm to IWD-CQ. Moreover, SA was chosen as a sample of well-known non-heuristic probabilistic method for global optimization problems. SOM, known as an effective tool for clustering problems, was also chosen as a representative of Artificial Neural Networks. The settings of tested AI methods are given bellow. All settings were tuned empirically to get the best feasible results.

- **IWD-CQ:** The number of IWDs is equal to the range of gray levels in the given image. Maximum number of the iterations was set to 2500 with a soil re initialization mechanism triggered every 30 iterations. Other settings of IWD-CQ are described in details in Section 3.
- **ACO:** Parameters α and β were set to 1 and pheromone evaporation coefficient ρ was set to 0.2. The number of ants is equal to the range of gray levels in the given image. Maximum number of the iterations was set to 2500 with a pheromone re initialization mechanism triggered every 50 iterations. Initial pheromone on the paths and Q were set to 50. Moreover, the elitism

mechanism was utilized and one of the best heuristics of IWD-CQ (Pop-HUD) was injected into the algorithm.

- **SA:** In the employed SA algorithm, initial temperature was set to 100, exponential update method was chosen to update the temperature and annealing function was set to fast annealing with a re-annealing interval of 100. The algorithm was allowed to run until the same number of function evaluations as IWD-CQ was achieved.
- **SOM:** Initial and final learning rates were set to 0.1 and 0.01 respectively. Initial and final neighborhood radii were chosen as √10 and 1, respectively. Maximum number of iterations was set to 2500.

The experiments were carried out using four different test images; Lena, Peppers, Baboon and House which are shown in Figure 4.1 (Full size images can be found in the Appendix). Different properties of each test image help to evaluate the performance of the algorithms under different conditions. Peppers image mainly consists of areas with smooth transitions and gradients. Baboon image, on the contrary, has a lot of details and high frequency areas. Lena image provides a combination of flat regions, detailed regions, texture and shading. House image contains several large and almost flat areas. The images are used in gray-scale with 256×256 pixel dimensions and they are quantized to eight gray levels. The quantization performance of each method is measured with Mean Absolute Error (MAE) defined by Equation 3.4.

a.  b. 

Figure 4.1. Test images; a) Lena; b) Peppers; c) Baboon; d) House (continued…)

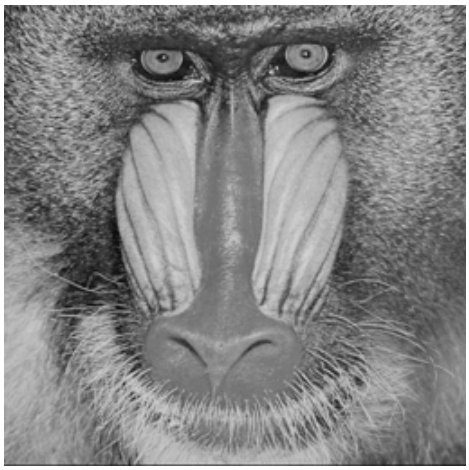
c.  d. 

Figure 4.1. (…continued) Test images; a) Lena; b) Peppers; c) Baboon; d) House

The examined conventional methods except Km-CQ always produce the same results for an image each time they are executed. However, AI methods, including IWD-CQ, may display different performances because of the randomness in their behavior. Also, Km-CQ has a random behavior in the initialization of centroids. In order to decrease the effect of randomness and achieve more robust outcomes, the results of the AI methods and Km-CQ are provided over fifteen independent runs of the algorithms. Arithmetic mean and standard deviation of the MAE values obtained from the tests along with the best results are presented in Tables 4.1-2.

The arithmetic means which form Average Performance Group display the central tendency of the obtained MAE values. On the other hand, the best results which form Best Performance Group show the minimum quantization error achieved in 15 runs. Consideration of the both performance groups helps to observe and compare the average and best achievable results of the algorithms.

Sample quantization results for Peppers image are shown in Figure 4.2. For Km-CQ, ACO, SA, SOM and IWD-CQ algorithms which have 15 results for each test image, the nearest result to the correspondent average MAE value is chosen for demonstration in Figure 4.2. Full size images can be found in the Appendix.

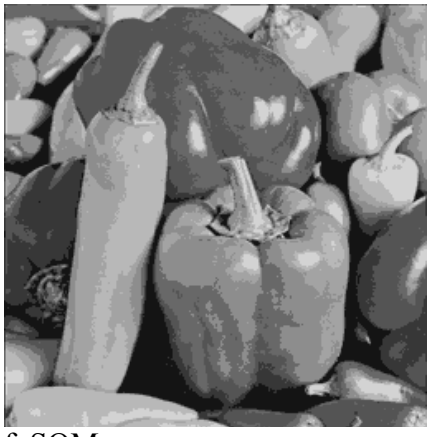
a. P-CQ

b. U-CQ

c. O-CQ

d. MC-CQ

e. Km-CQ



f. SOM

Figure 4.2 Sample quantization results of the algorithms for Peppers image. All the images are quantized to eight gray-levels. It can be seen that P-CQ and P-M-HUD produce the worst and the best quality. Full size images can be found in the Appendix. (continued…)

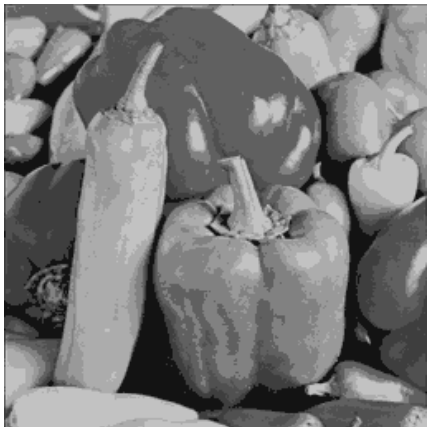
g. ACO

h. SA

i. Con-HUD

j. Pop-HUD

k. Med-HUD



l. P-M-HUD

Figure 4.2 (…continued) Sample quantization results of the algorithms for Peppers image. All the images are quantized to eight gray-levels. It can be seen that P-CQ and P-M-HUD produce the worst and the best quality. Full size images can be found in the Appendix.

Table 4.1. MAE values of the algorithms for Lena and Peppers images.

	Lena			Peppers		
Method	Best	Average	STD	Best	Average	STD
P-CQ	10.7158	-	-	13.5394	-	-
U-CQ	9.6189	-	-	9.9121	-	-
O-CQ	8.0627	-	-	7.8084	-	-
MC-CQ	5.8430	-	-	7.6074	-	-
Km-CQ	5.8419	6.7293	0.6155	6.8322	7.5324	0.4540
SOM	5.5015	5.9263	0.3022	6.4358	6.6066	0.1501
ACO	5.7857	5.9320	0.0777	6.5878	6.7166	0.0789
SA	5.7357	6.6700	0.3603	6.7400	7.2911	0.3675
IWD-CQ Con-HUD	5.6119	5.8111	0.1391	6.4094	6.5827	0.1072
IWD-CQ Pop-HUD	5.4367*	5.6379*	0.0954	6.2336*	6.4727	0.1139
IWD-CQ Med-HUD	5.5100	5.6803	0.1259	6.3122	6.4598	0.1008
IWD-CQ P-M-HUD	5.4577	5.6908	0.1359	6.2370	6.4570*	0.1164

* least MAE values

Table 4.2. MAE values of the algorithms for Baboon and House images.

	Baboon			House		
Method	Best	Average	STD	Best	Average	STD
P-CQ	12.5567	-	-	6.5602	-	-
U-CQ	9.2628	-	-	8.8390	-	-
O-CQ	6.2305	-	-	9.0188	-	-
MC-CQ	6.0728	-	-	6.2420	-	-
Km-CQ	5.8607	6.3508	0.4887	4.1720	5.3048	0.6336
SOM	5.4799	5.8162	0.2420	4.0822	4.3628	0.2527
ACO	5.4138	5.5204	0.0681	4.0228	4.2362	0.0935
SA	5.6924	6.4286	0.3351	4.4118	4.9694	0.2249
IWD-CQ Con-HUD	5.2566	5.4503	0.1132	3.9196	4.0938	0.0755
IWD-CQ Pop-HUD	5.0211	5.2748	0.1381	3.8065*	4.0264	0.1056
IWD-CQ Med-HUD	5.0140*	5.2320*	0.1129	3.8338	4.0081	0.0958
IWD-CQ P-M-HUD	5.0912	5.2705	0.1404	3.8131	3.9624*	0.0987

* least MAE values

The results show that the algorithms tested may display diverse performances under different performance measures, i.e.; “Best” and “Average”. For example, while SOM has the third better performance for Lena image among the best MAE values, it has the sixth better performance in average MAE values for the same image. Therefore the discussions should be made separately for the “Average” and the “Best” MAE values. Meanwhile, the differences between quantization

performances of some of the algorithms for the same image are only fractions of one in a thousandth. IWD-CQ with P-M-HUD, for example, noted as the best performing algorithm in the average results group with 6.4570 MAE value for the Peppers image. Med-HUD heuristic is, on the other hand, runner up for the same group with slightly worse MAE value which is only 0.0028 more than P-M-HUD's MAE value. Therefore, a scoring mechanism was required to be devised to evaluate an algorithm's performance relative to the other algorithms' performances. The scores were calculated for twenty four subgroups that are formed as follows:

Three major comparison groups were formed

- between IWD-CQ`s heuristics.
- between IWD-CQ and conventional methods.
- between IWD-CQ and AI methods.

Each of these major groups was divided into two performance groups; the best and the average. Finally, the scores were calculated independently for each test image which provided twenty four subgroups (3 major groups × 2 performance groups × 4 test images). Assuming that MAE_{max} and MAE_{min} are respectively the maximum and minimum of MAE values inside a subgroup, the scores are calculated by the following equation.

$$Score = \frac{MAE_{max} - MAE}{MAE_{max} - MAE_{min}} \tag{4.1.}$$

Using Equation 4.1, the scores of maximum and minimum MAE values inside a subgroup are assigned to zero and one, respectively. The remaining MAE values get a score based on their relative quality. By adding the scores of an algorithm, obtained from different images, an overall performance measure can be calculated separately for performance groups "best" and "average". The overall performances are indicated by *Total* in Tables 4.3-5. This allows us to draw a clearer conclusion out of the results and to get an overall outcome about the performances of the algorithms.

4.1.1. Quantization performance of the heuristic methods

It is a well-known fact that an appropriate local heuristic can play an important role for the success of an optimization algorithm. Four different local heuristic methods are introduced to investigate their influence on the performance of the proposed IWD-CQ algorithm. Detailed results displaying the best and average MAE values of each heuristic method over 15 independent runs can be found in Tables 4.1-2. The first major comparison group was formed from the outcomes of the four heuristics of IWD-CQ algorithm presented in Tables 4.1-2. The scores for the heuristics were, then, calculated for each test image in each performance group by using Equation 4.1. The scores for the heuristics of IWD-CQ algorithm that were calculated for eight subgroups are given in Table 4.3.

Table 4.3. Scores for the heuristics of IWD-CQ algorithm.

	Best Performance Group					Average Performance Group				
Method	Lena	Peppers	Baboon	House	*Total*	Lena	Peppers	Baboon	House	*Total*
Con-HUD	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Pop-HUD	1.0000*	1.0000*	0.9706	1.0000*	3.9706*	1.0000*	0.8748	0.8040	0.5127	3.1915
Med-HUD	0.5818	0.5529	1.0000*	0.7586	2.8933	0.7551	0.9776	1.0000*	0.6520	3.3847
P-M-HUD	0.8801	0.9806	0.6816	0.9421	3.4845	0.6947	1.0000*	0.8235	1.0000*	3.5182*

* highest scores

Table 4.3 clearly demonstrates that assignment of a constant value for the heuristic does not yield good results. When Con-HUD is compared with other heuristics, it acquires zero score for all the images in all performance groups. Therefore Con-HUD is not recommended to be used as an appropriate heuristic for IWD-CQ and it will not be included in the rest of discussions in this section.

In the best performance group, Pop-HUD shows an outstanding performance acquiring the highest scores for Lena, Peppers and House images. For Baboon image, Pop-HUD obtains the second better score with a value of 0.9706 which is very close to the highest score that is obtained by Med-HUD. In total, Pop-HUD acquires 3.9706. This shows the overall superiority of the Pop-HUD heuristic in the best performance group.

Considering the average performance group, it can be seen that performance of the three heuristics are dependent on the properties of test images. P-M-HUD acquires highest scores in images having a lot of gradients and flat regions like Peppers and House. On the contrary, Med-HUD shows a significant performance for Baboon image attaining the highest scores in all performance groups. This can exemplify the overall superiority of Med-HUD in high frequency images. On the other hand, Pop-HUD acquires the highest scores in all performance groups for Lena image which is a combination of high frequency, gradient and flat areas. Overall, P-M-HUD obtains the highest total score 3.5182 in average performance group.

4.1.2. Comparison of IWD-CQ with conventional methods

In this section, the quantization performances of IWD heuristics (Con-HUD, Pop-HUD, Med-HUD and P-M-HUD) are compared with performances of the conventional methods. The second major comparison group was formed from the outcomes of conventional methods and IWD-CQ heuristics. The scores were calculated for each image in each performance group and given in Table 4.4.

Table 4.4. Scores for the IWD-CQ heuristics and conventional methods.

	Best Performance Group					Average Performance Group				
Method	Lena	Peppers	Baboon	House	*Total*	Lena	Peppers	Baboon	House	*Total*
IWD-CQ Con-HUD	0.9668	0.9759	0.9678	0.9783	3.8889	0.9659	0.9823	0.9702	0.9740	3.8923
IWD-CQ Pop-HUD	1.0000*	1.0000*	0.9991	1.0000*	3.9991*	1.0000*	0.9978	0.9942	0.9873	3.9793
IWD-CQ Med-HUD	0.9861	0.9892	1.0000*	0.9948	3.9701	0.9916	0.9996	1.0000*	0.9910	3.9822
IWD-CQ P-M-HUD	0.9960	0.9995	0.9898	0.9987	3.9841	0.9896	1.0000*	0.9947	1.0000*	3.9843*
P-CQ	0.0000	0.0000	0.0000	0.4717	0.4717	0.0000	0.0000	0.0000	0.4862	0.4862
U-CQ	0.2078	0.4965	0.4367	0.0345	1.1755	0.2160	0.5122	0.4497	0.0356	1.2134
O-CQ	0.5026	0.7844	0.8387	0.0000	2.1257	0.5225	0.8092	0.8637	0.0000	2.1953
MC-CQ	0.9230	0.8120	0.8596	0.5327	3.1274	0.9596	0.8376	0.8852	0.5492	3.2315
Km-CQ	0.9232	0.9181	0.8877	0.9299	3.6589	0.7851	0.8482	0.8473	0.7345	3.2150

* highest scores

Table 4.4, clearly demonstrates that, amongst the conventional color quantization methods P-CQ has the least scores in all performance groups in all test images except House image. It acquires the lowest total scores 0.4717 and 0.4862 in best and average performance groups, respectively. MC-CQ and Km-CQ displays

better overall performances than other conventional methods by obtaining scores more than three in both performance groups. However it should be noted that Pop-HUD outperforms all the remaining algorithms by obtaining 3.9991 out of 4 in total scores for the best performance group. In the average performance group, P-M-HUD gets the highest overall score, 3.9843. None of the conventional methods obtains better scores than any of the IWD heuristics within second major comparison group and it can be concluded that IWD heuristics overwhelmingly outperforms the conventional methods.

4.1.3. Comparison of IWD-CQ with Artificial Intelligence methods

The third major comparison group was formed from the results of IWD-CQ heuristics and AI methods; Self Organizing Maps (SOM), Ant Colony Optimization (ACO) and Simulated Annealing (SA). The scores in Table 4.5 were calculated as described in the previous sections.

Table 4.5. Scores for the IWD-CQ heuristics and AI methods.

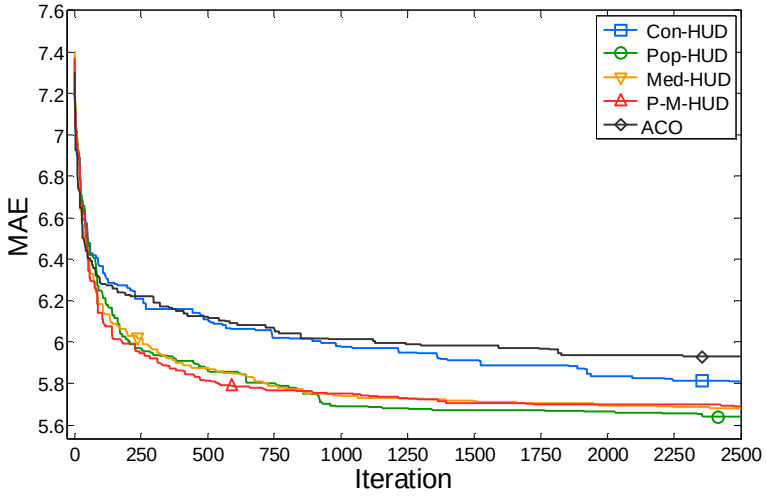
	Best Performance Group					Average Performance Group				
Method	Lena	Peppers	Baboon	House	*Total*	Lena	Peppers	Baboon	House	*Total*
IWD-CQ Con-HUD	0.4981	0.6528	0.6423	0.8132	2.6065	0.8322	0.8493	0.8175	0.8694	3.3684
IWD-CQ Pop-HUD	1.0000*	1.0000*	0.9895	1.0000*	3.9895*	1.0000*	0.9811	0.9642	0.9364	3.8817
IWD-CQ Med-HUD	0.7901	0.8448	1.0000*	0.9549	3.5898	0.9589	0.9966	1.0000*	0.9546	3.9101
IWD-CQ P-M-HUD	0.9398	0.9933	0.8861	0.9892	3.8084	0.9488	1.0000*	0.9678	1.0000*	3.9166*
SOM	0.8144	0.6008	0.3132	0.5446	2.2730	0.7205	0.8207	0.5118	0.6023	2.6553
ACO	0.0000	0.3006	0.4106	0.6426	1.3539	0.7150	0.6888	0.7589	0.7281	2.8908
SA	0.1434	0.0000	0.0000	0.0000	0.1434	0.0000	0.0000	0.0000	0.0000	0.0000

* highest scores

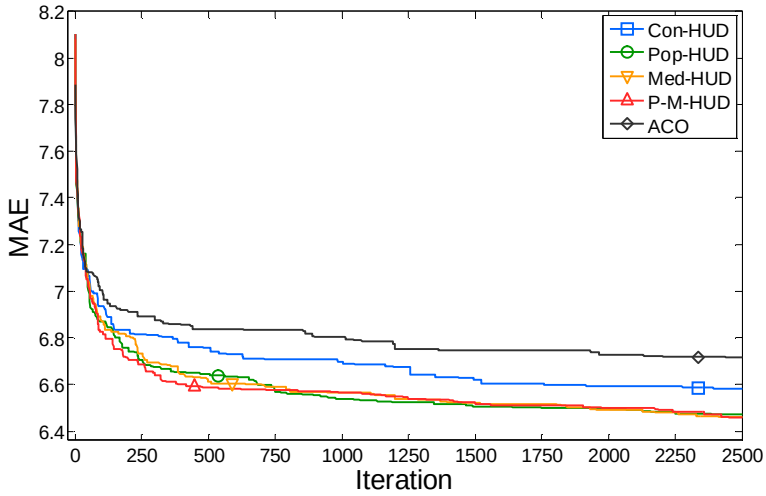
The results in Table 4.5 show that SA is the worst performing AI algorithm within this major comparison group. It gathers the lowest total scores in both performance groups. Moreover, it can be seen that the total score of ACO in best performance group is significantly lower than IWD-CQ heuristics and SOM. However, ACO acquires slightly higher score than SOM in average performance group. Nevertheless, IWD-CQ heuristics acquire higher scores than all the AI methods in both performance groups.

4.1.4. Convergence properties

It is proven that IWD algorithm is able to find the optimal solution of the problem if it continues to run for a sufficient number of iterations (Shah-Hosseini, 2008). Figure 4.3 illustrates the convergence diagrams of IWD-CQ with its four heuristics. In addition, convergence diagrams of ACO have been added into figures for comparison.



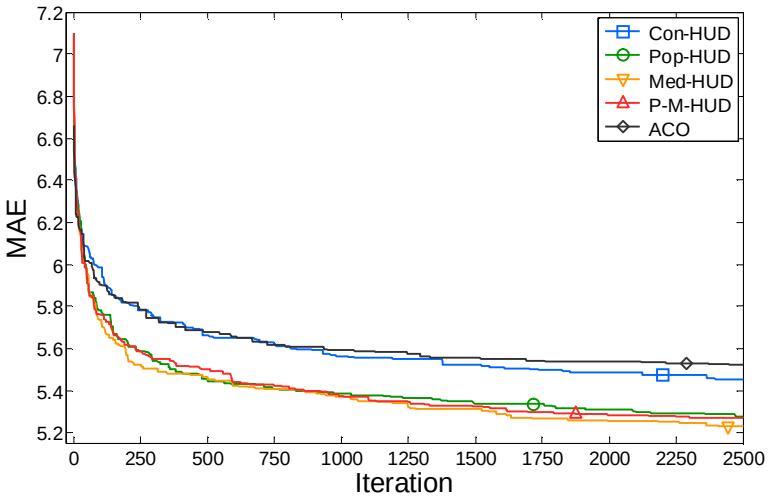
a.



b.

Figure 4.3. Convergence diagrams of IWD-CQ and ACO showing the average of MAE values over 15 runs for a) Lena, b) Peppers, c) Baboon, and d) House images. (continued…)

c.



d.

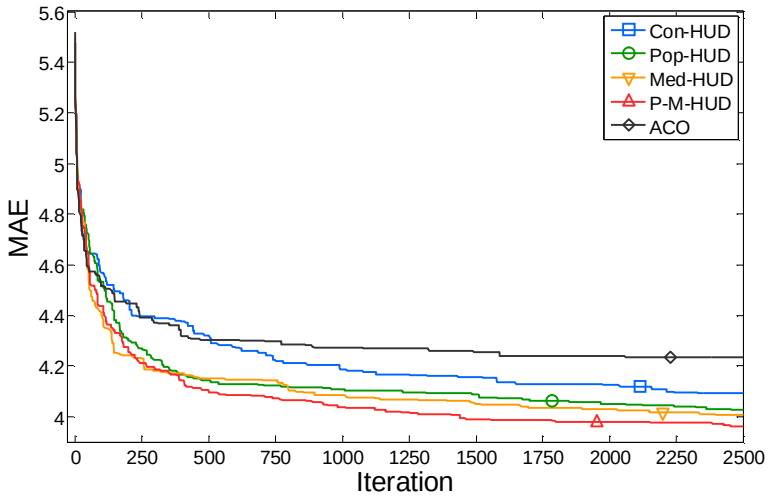


Figure 4.3. (…continued) Convergence diagrams of IWD-CQ and ACO showing the average of MAE values over 15 runs for a) Lena, b) Peppers, c) Baboon, and d) House images.

The convergence diagrams illustrate that IWD-CQ converges quickly to a good solution in 500 iterations. More importantly, it can be seen that after the significant convergence in the early iterations, the algorithm does not get stuck in local minima and continues to converge in a slower pace. This is a promising characteristic for the proposed algorithm which hopefully will take it towards finding the global minimum. The time needed for converging to the global optimum can be reduced by fine tuning of the parameters (Shah-Hosseini, 2008). Moreover, the diagrams clearly show the weak performance of Con-HUD heuristic in all images. This indicates that an inappropriate heuristic is considerably able to affect the results of a metaheuristic algorithm.

The diagrams, also, show that even the worst performing heuristic of IWD-CQ, Con-HUD, performs better than ACO in all the test images. It can be seen that approximately after 1000th iteration convergence curves of ACO algorithm become almost flat. Although IWD-CQ and ACO perform almost the same in early iterations, IWD-CQ behaves clearly better in the next iterations whilst ACO suffers from getting stuck in local minima.

Formerly, in Section 4.1.1, P-M-HUD was shown as the best performing heuristic in average performance group. In this regard, it exhibits the highest quantization quality for Peppers and House images. For Baboon and Lena images, it displays the second and third better performances. In Figure 4.4, convergence curve of IWD-CQ with P-M-HUD heuristic is shown for Lena image. The figure indicates the iteration numbers that the algorithm reaches to quantization quality of the other algorithms. The surpassing points indicated in Figure 4.4 are based on the final MAE values of the algorithms in average performance group which can be found in Tables 4.1-2.

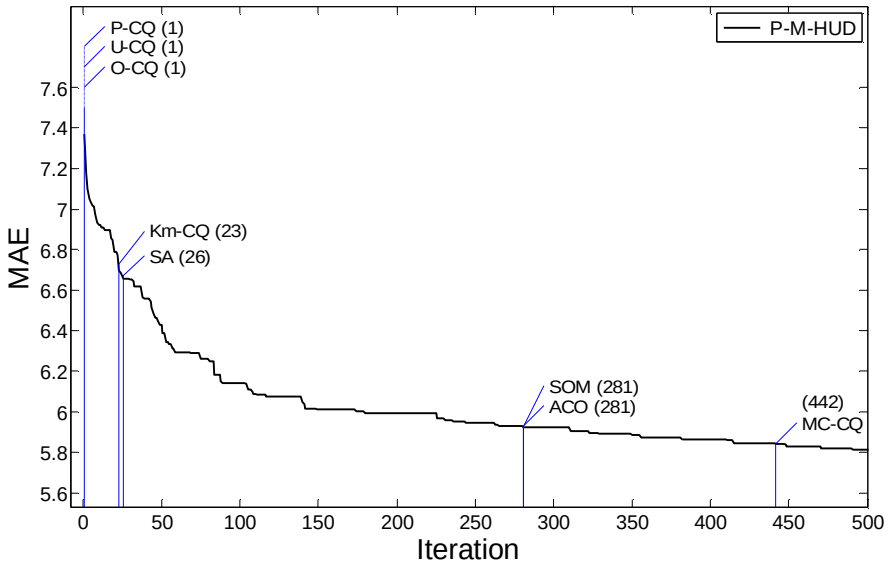


Figure 4.4. Convergence diagram of IWD-CQ with P-M-HUD for Lena image, the numbers in parenthesis indicate the iteration number that P-M-HUD surpasses the correspondent algorithm.

It can be seen that IWD-CQ surpasses all the algorithms before its 500th iteration. In the first iteration of the algorithm, P-CQ, U-CQ and O-CQ are surpassed.

Then, the algorithm surpasses Km-CQ, SA, SOM, ACO and MC-CQ in iterations 23, 26, 281, 281 and 442 respectively. It can be concluded that the proposed algorithm is able to provide promising results even if it is executed in a low number of iterations.

4.2. DE-PCQ

A comparative work has been carried out in order to evaluate the performance of the proposed DE-PCQ algorithm in terms of perceptual quality of the quantized image. The comparisons were made against five commonly used CQ algorithms including; Uniform CQ (U-CQ), Popularity CQ (P-CQ) (Boyle and Lippman, 1978), Median-Cut CQ (MC-CQ) (Heckbert, 1980), Octree CQ (O-CQ) (Gervautz and Purgathofer, 1988) and K-means CQ (Km-CQ) (Lloyd, 1957).

The same test images as Section 4.1; Lena, Peppers, Baboon and House are used in the experiments. However, unlike Section 4.1, the images are provided in colors being converted to the CIELab color space. The images are used in 256×256 pixel dimensions and they are quantized to eight colors. The quantization performance of each method is measured with ΔE defined by Equation 3.6. Figure 4.5 presents the used test images. Full size images can be found in the Appendix.

a.  b. 

Figure 4.5. Test images; a) Lena; b) Peppers; c) Baboon; d) House (continued…)

c.  d. 

Figure 4.5. (…continued) Test images; a) Lena; b) Peppers; c) Baboon; d) House

Standard variant of the DE algorithm is chosen in this section to implement the proposed perceptual color quantization method. This selection is made according to the fact that basic DE variants with properly tuned parameters have comparable performances (Pedersen, 2010a; Pedersen and Chipperfield, 2011).

4.2.1. Parameter Tuning

Proper tuning of the three parameters F, CR, and N_P plays an important role on optimization performance of DE algorithm (Equations 3.1-2). Parameter N_P specifies the size of population ($N_P \geq 4$) and parameters F and CR determine differential weight and crossover probability respectively. Although some recent studies aim to develop adaptive parameter tuning methods for DE, these algorithms only bring up new parameters that must be set by the user (Pedersen and Chipperfield, 2008; Pedersen, 2010b). Therefore, a non-adaptive approach is used for parameter tuning.

In this study, N_P is set to 240 based on the Equation 3.7 suggested by Storn (1996). Additionally, In order to determine the proper values of F and CR, performance landscapes of the proposed algorithm are calculated for F and CR. Although value of F varies in the range [0,2], the performance landscape is calculated for the range [0.1,1.1]. This is based on the fact that values out of this range rarely produce optimal results. The research carried out by Pedersen (2010b)

shows that for different problem configurations, in almost all the cases, optimal value of *F* falls into the range [0.1,1.1]. Based on a similar discussion, value of *CR* is limited to the range [0.4,0.95].

Figures 4.6-7 shows the performance landscapes for *objFuncBCQ* and *objFuncPCQ* respectively which are computed using Lena as the test image. Because of randomness in the behavior of DE-PCQ, the values forming the performance landscapes are provided by averaging over 15 runs for each combination to decreases the effect of randomness.

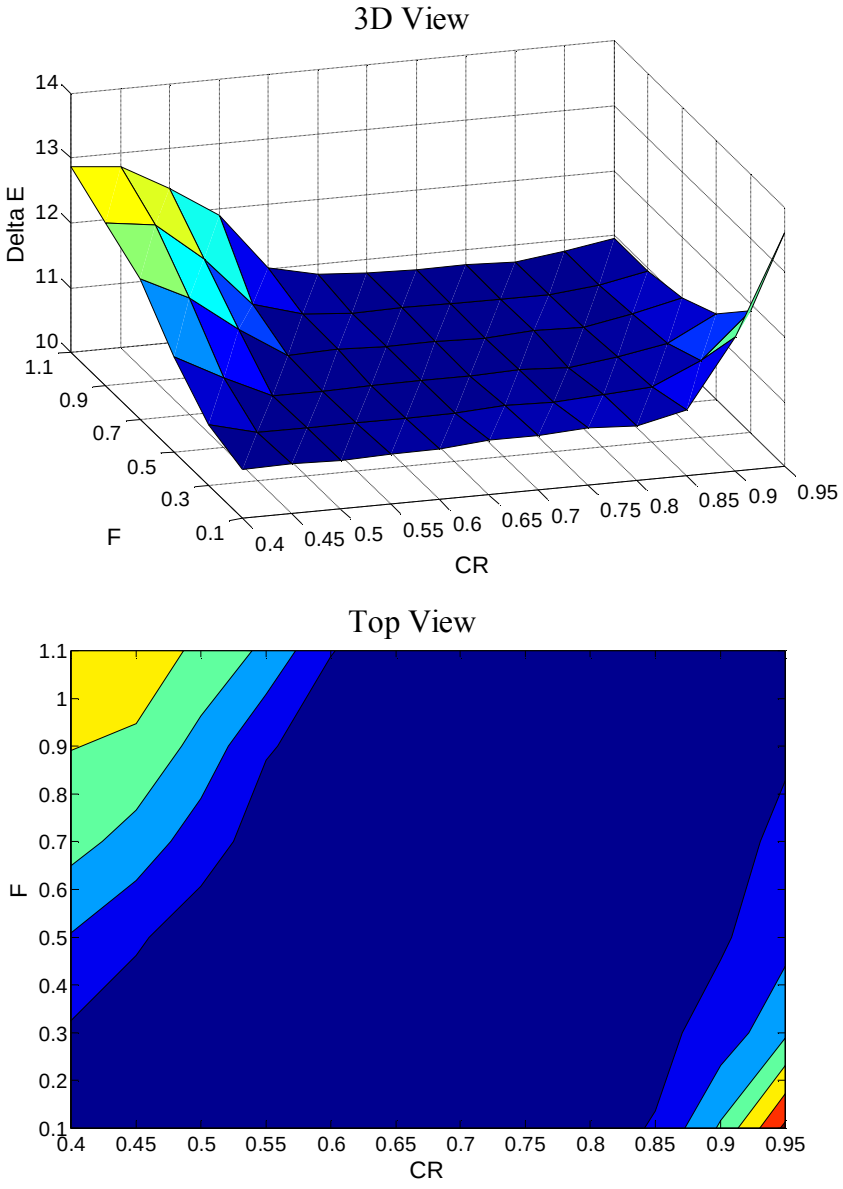


Figure 4.6. Performance Landscape for *objFuncBCQ*.

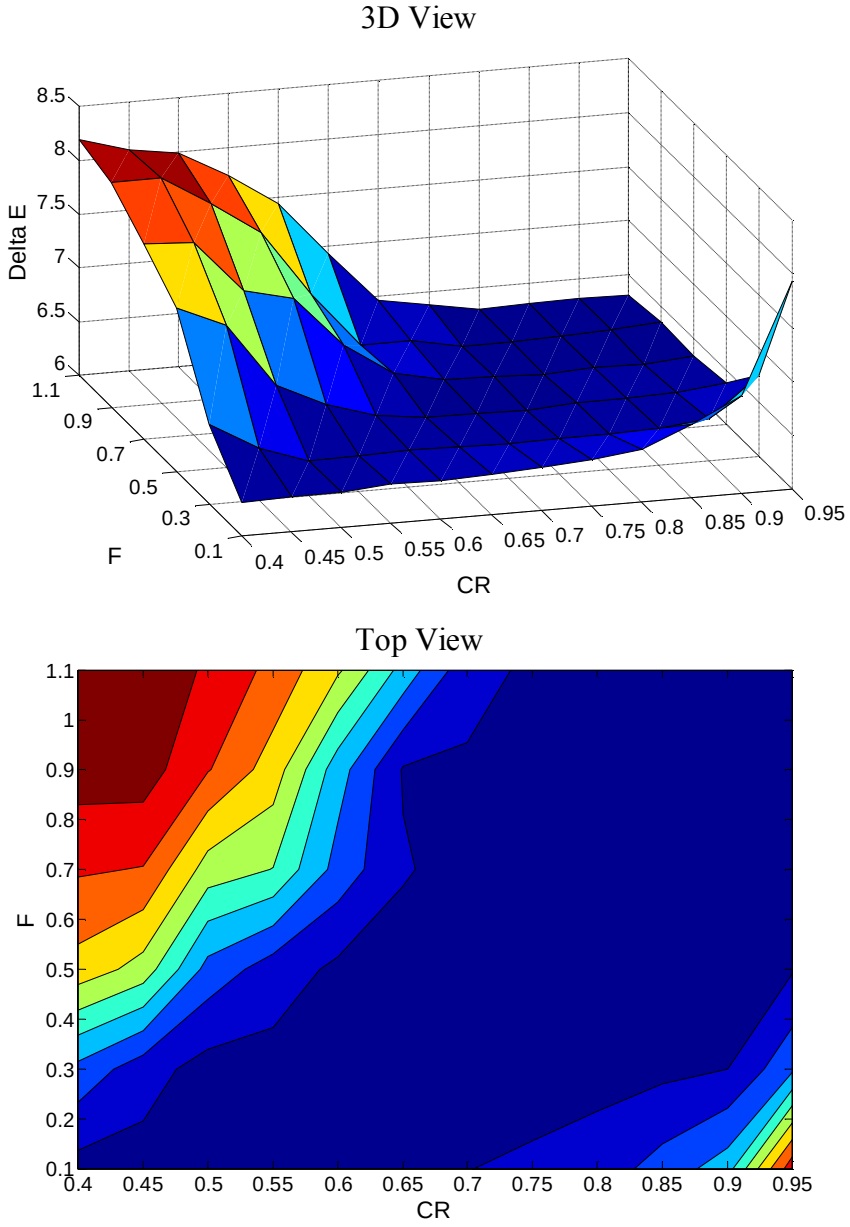


Figure 4.7. Performance Landscape for *objFuncPCQ*.

Performance landscapes show that there is an approximate linear proportionality between *CR* and *F* for optimal results. More specifically, the higher values of *F* demand higher values of *CR* in order to obtain optimal results. Moreover it can be seen that there is a broad range of combinations for *F* and *CR* yielding in optimal results. This range is wider for the objective function of Basic CQ; *objFuncBCQ*.

The best values of *F* and *CR* for the both objective functions are presented in the following table.

Table 4.6. Optimal values of *CR* and *F* for DE-PCQ algorithm.

Objective Function	*F*	*CR*	Min ΔE
objFuncBCQ	0.5	0.55	10.7294
objFuncPCQ	0.7	0.8	6.2878

4.2.2. Quantization performance and comparison

This section presents the quantization performance of DE-PCQ. In addition, results obtained by the similar perceptual approach for five commonly used CQ methods; U-CQ, P-CQ, O-CQ, MC-CQ and Km-CQ are provided for comparison. Regarding that DE-PCQ and Km-CQ show different performances because of the randomness in their behavior, their results are provided over 15 independent runs of the algorithms. Arithmetic mean and standard deviation of the ΔE values obtained from the experiments along with the best results are presented in Tables 4.7-8.

"Method (Halftoning)" in the tables implies that after determination of the final palette, the quantization result is prepared using error diffusion halftoning. However, "Method (No Halftoning)" implies that after determination of the final palette, the quantization result is prepared without halftoning. Nonetheless, the Gaussian filter is applied in both the cases in order to simulate the averaging property of HVS.

Table 4.7. ΔE values of the algorithms for Lena and Peppers images.

	Lena			Peppers		
Method (Halftoning)	Best	Mean	STD	Best	Mean	STD
U-CQ	100.089	-	-	97.984	-	-
P-CQ	23.757	-	-	27.949	-	-
O-CQ	17.566	-	-	38.240	-	-
MC-CQ	16.359	-	-	22.354	-	-
Km-CQ	8.207	8.427	0.070	9.756	10.193	0.377
DE-PCQ *objFuncBCQ*	7.818	8.058	0.146	9.338	9.520	0.195
DE-PCQ *objFuncPCQ*	6.271*	6.288*	0.012	7.781*	7.810*	0.017
Method (No Halftoning)						
U-CQ	136.919	-	-	135.878	-	-
P-CQ	25.202	-	-	35.767	-	-
O-CQ	21.072	-	-	41.501	-	-
MC-CQ	18.481	-	-	25.945	-	-
Km-CQ	9.377	9.904	0.278	11.176	11.671	0.334
DE-PCQ *objFuncBCQ*	8.514	8.777	0.109	10.540	10.624	0.142
DE-PCQ *objFuncPCQ*	10.396	11.123	0.526	11.973	13.007	0.541

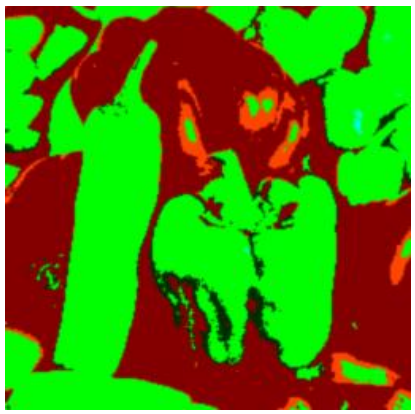
* least ΔE values

Table 4.8. ΔE values of the algorithms for Baboon and House images.

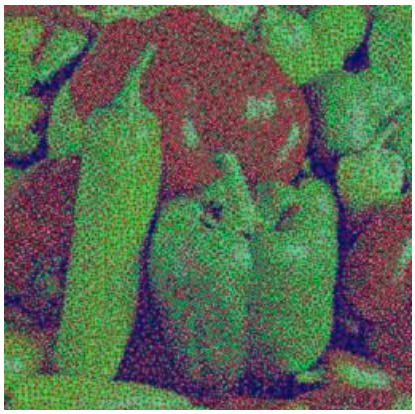
	Baboon			House		
Method (Halftoning)	Best	Mean	STD	Best	Mean	STD
U-CQ	86.366	-	-	92.827	-	-
P-CQ	67.782	-	-	25.540	-	-
O-CQ	30.884	-	-	16.963	-	-
MC-CQ	27.769	-	-	12.933	-	-
Km-CQ	11.034	11.142	0.151	4.850	5.363	0.715
DE-PCQ *objFuncBCQ*	10.991	11.275	0.219	4.927	5.066	0.086
DE-PCQ *objFuncPCQ*	9.238*	9.265*	0.024	4.469*	4.540*	0.040
Method (No Halftoning)						
U-CQ	132.681	-	-	145.607	-	-
P-CQ	83.366	-	-	24.621	-	-
O-CQ	33.425	-	-	17.793	-	-
MC-CQ	29.284	-	-	16.082	-	-
Km-CQ	12.038	12.519	0.369	5.441	6.143	0.589
DE-PCQ *objFuncBCQ*	11.781	12.019	0.226	5.011	5.121	0.049
DE-PCQ *objFuncPCQ*	12.860	13.172	0.204	5.343	5.640	0.268

* least ΔE values

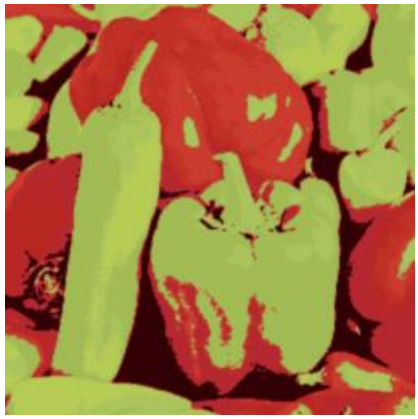
Sample quantization results for Peppers image are shown in Figure 4.8 (Full size images can be found in the Appendix). For Km-CQ and DE-PCQ algorithms which have 15 results for each test image, the nearest result to average ΔE value is chosen to be shown in the figure.



a. U-CQ (No Halftoning)

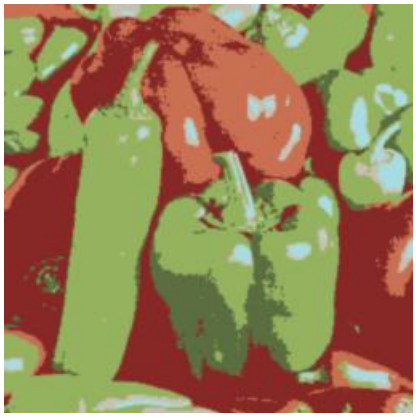


b. U-CQ (Halftoning)



c. P-CQ (No Halftoning)

d. P-CQ (Halftoning)



e. O-CQ (No Halftoning)

f. O-CQ (Halftoning)

Figure 4.8. Sample quantization results of the algorithms for Peppers image. (continued…)

g. MC-CQ (No Halftoning)

h. MC-CQ (Halftoning)

i. Km-CQ (No Halftoning)

j. Km-CQ (Halftoning)

k. DE-PCQ *objFuncBCQ* (No Halftoning)

l. DE-PCQ *objFuncBCQ* (Halftoning)

Figure 4.8. (…continued) Sample quantization results of the algorithms for Peppers image. (continued…)

m. DE-PCQ *objFuncPCQ* (No Halftoning)

n. DE-PCQ *objFuncPCQ* (Halftoning)

Figure 4.8 (…continued) Sample quantization results of the algorithms for Peppers image. The images are quantized to 8 colors. It can be seen that U-CQ and DE-PCQ *objFuncPCQ* (Halftoning) produce the worst and the best quality.

Considering quantization performance of the conventional algorithms, it can be seen that U-CQ produces the worst and very low quality results in CIELab color space (Tables 4.7-8). P-CQ produces the second worst results for all the images except for Peppers image which O-CQ owns the second worst performance for it. Remaining conventional algorithms except Km-CQ produce relatively same results for all images. Amongst conventional methods, Km-CQ produces the best results which are significantly better than the results of other conventional algorithms.

Moreover, comparison results of the conventional methods considering "Halftoning" and "No Halftoning" shows that lower perceptual quantization errors are achieved with "Halftoning". This is true for all the cases ignoring slightly worse performance of P-CQ for House image. The mentioned fact demonstrates that halftoning is an effective method for producing perceptually higher quality images.

Considering quantization performance of DE-PCQ, the results presented in Tables 4.8-9 and Figure 4.8 show that *objFuncPCQ* with "Halftoning" produces the least quantization error amongst all the tested methods. Comparison results of the two objective functions with "Halftoning" and "No Halftoning" reveals that *objFuncPCQ* performs better than *objFuncBCQ* when halftoning is applied for the final quantization process. In contrary, *objFuncBCQ* performs better when halftoning is not applied ("No Halftoning"). It should be noted that here, "Halftoning" and "No

Halftoning" indicates the halftoning process applied to the final quantization process using the optimal palette that is obtained beforehand. In other words, the optimal palette may be obtained with or without halftoning independently. The mentioned result is based on the fact that *objFuncPCQ* optimizes the palette applying halftoning technique and Gaussian filter. Thus, it produces the best result when halftoning, also, is applied for the final quantization process. On the other hand, *objFuncBCQ* optimizes the palette in a basic CQ process. Therefore, it performs better when halftoning is not applied during the final quantization process. It should be noted that in the both "Halftoning" and "No Halftoning" methods, Gaussian filter is applied in order to simulate the averaging property of human eye.

The aforementioned discussion shows the effectiveness of the idea behind the proposed perceptual approach. ΔE values in Tables 4.7-8 show that DE-PCQ with *objFuncPCQ* and "Halftoning" acquires %25, %23, %17 and %15 lower quantization errors than the best conventional method, Km-CQ, for Lena, Peppers, Baboon, and House images respectively. The percentages are calculated considering Mean ΔE values. Moreover, this result can be seen clearly from the quantized images shown in Figure 4.8.

Additionally, comparison of *objFuncBCQ* with "No Halftoning" against *objFuncPCQ* with "Halftoning" isolates the effect of introduced perceptual approach using DE algorithm as an AI based optimizer. In this regard, it can be seen that %28, %26, %23 and %11 lower quantization errors are achieved in *objFuncPCQ* with "Halftoning" for Lena, Peppers, Baboon, and House images respectively. The percentages are again calculated considering Mean ΔE values presented in Tables 4.7-8. Therefore, *objFuncPCQ* with "Halftoning" is empirically proven to be effective and recommended as a result of the discussion.

4.2.3. Convergence properties

Differential Evolution algorithm is known as a fast optimization algorithm (Price et al., 2005). Convergence diagrams in Figures 4.9-10 shows that the DE-PCQ algorithm performs its major convergence in the first 300 iterations. Then, it continues to converge in a slower pace and finally stops approximately after 600th iteration.

The diagrams show that convergence of DE algorithm for *objFuncPCQ* is quicker than the convergence of *objFuncBCQ*. A possible reason can be the difference between the optimal parameter values that are used for the two objective functions. It is a clear fact that performance and convergence of DE algorithm can be affected by *F* and *CR* values. In this study the mentioned values are tuned for obtaining lower objective function values.

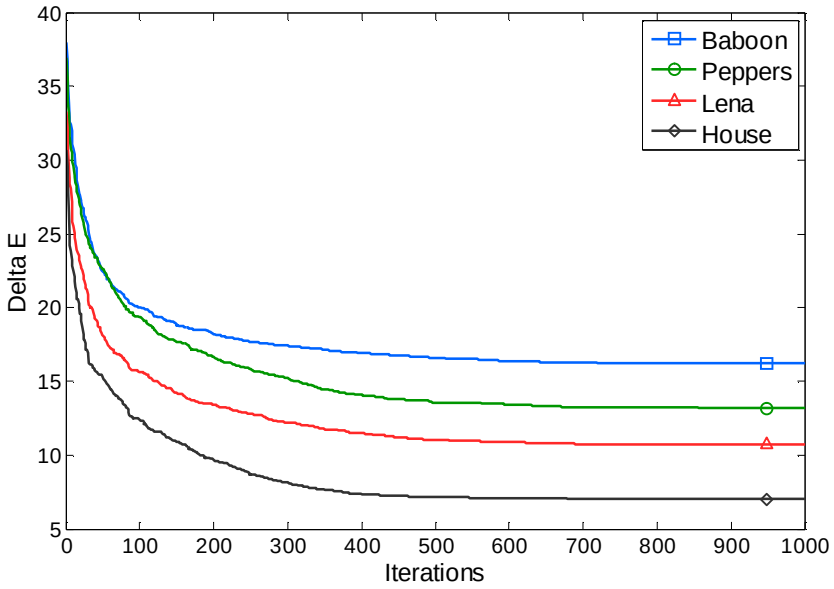


Figure 4.9. Convergence diagram of DE-PCQ algorithm for objective function of Basic CQ; *objFuncBCQ*.

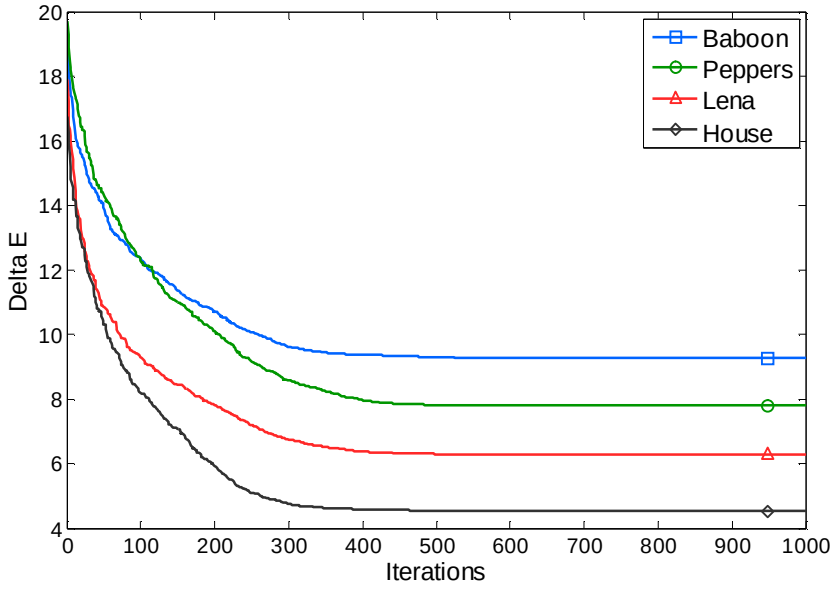


Figure 4.10. Convergence diagram of DE-PCQ algorithm for objective function of Perceptual CQ; *objFuncPCQ*.

The convergence diagrams are plotted using the average ΔE values obtained from 15 independent runs of the algorithm. It can be seen that the recommended objective function, *objFuncPCQ*, obtains obviously lower perceptual quantization errors than *objFuncBCQ*. Moreover it converges quicker than *objFuncBCQ* which does not employ the proposed perceptual approach. The convergence diagram shown in Figure 4.10 demonstrates that even in 300 iterations, effective and promising results can be obtained which makes the algorithm appropriate to be used in time constraint applications.

5. CONCLUSIONS

5.1. IWD-CQ

In the first part of the study, a novel swarm-intelligence-based approach, IWD-CQ was presented for Color Quantization (CQ) problem. In this regard, Intelligent Water Drops (IWD) algorithm, for the first time, was adapted to CQ problem. Moreover, four heuristics were introduced and utilized in the proposed algorithm to observe their effects on the quantization performance. The experiments were carried out using four test images; Lena, Peppers, Baboon and House. IWD-CQ heuristics were compared within themselves to choose the best heuristic and with other conventional and AI methods.

The comparisons within IWD-CQ heuristics revealed that choosing a constant value for the heuristic does not provide good results. The quantization performances of other heuristics are slightly content dependent. Med-HUD displays its highest performance for high frequency image Baboon. Pop-HUD acquires the highest scores for Lena image which is a combination of high frequency, gradient and flat areas. P-M-HUD, on the other hand, performs well in low frequency images. Overall, Pop-HUD and P-M-HUD were found as the best performing heuristics in best and average performance groups, respectively. The performances of IWD-CQ heuristics have been overwhelmingly better than the tested conventional and AI methods.

Although the computational cost of the proposed algorithm is relatively high, it was shown that the execution of the algorithm in low iterations is still effective. Therefore, the algorithm could be run in low iterations to produce reasonably good results in applications that demand constraint run-time. Meanwhile, longer executions of the algorithm provide much better results which can be used in offline applications or to produce optimal or near optimal palettes for benchmark use. For future studies, the algorithm may be extended to be applied for RGB images. Moreover, by fine tuning of the parameters, and/or by implementing better heuristics, better results may be obtained.

5.2. DE-PCQ

In the second part of the study a different CQ approach were investigated. A model of Human Visual System (HVS) was employed to obtain higher perceptual qualities. In this regard, Differential Evolution algorithm was implemented for the first time to minimize the perceptual quantization error using the mentioned model of HVS. The proposed algorithm DE-PCQ is based on the property of HVS in perceiving an average color of the adjacent pixels. Based on the fact that a uniform color space should be used for a perceptual CQ algorithm, CIELab color space was chosen to be employed and the quantization error was measured with a perceptually uniform quantization error; ΔE. The colored version of test images Lena, Peppers, Baboon and House were used to evaluate the quantization performance of DE-PCQ in term of perceptual quality of the quantized images.

Comparisons were carried out against five commonly used conventional CQ algorithms. The comparison results showed the significant superiority of the proposed algorithm in obtaining lower perceptual quantization errors. Moreover, two different objective functions were implemented in the algorithm and their behavior was observed and discussed. As the result, DE-PCQ with *objFuncPCQ* and "Halftoning" was recommended as an effective perceptual CQ method. Although execution cost of DE-PCQ, as an Artificial Intelligence method, is higher than the execution cost of conventional methods, it was observed that even running of the algorithm in low iterations produces reasonably good results.

Overall, the study showed the effectiveness of proposed algorithms; IWD-CQ and DE-PCQ in obtaining better quantization results than all the other tested algorithms. One of the prominent advantages of IWD-CQ and DE-PCQ was observed to be free from special types of artifacts that are common in conventional methods. This can be seen clearly from the quantized images shown in the thesis.

REFERENCES

ATKINS, C. B., FLOHR, T. J., HILGENBERG, D. P., BOUMAN, C. A., and ALLEBACH, J. P., 1994. Model-Based Color Image Sequence Quantization. Proceedings of the SPIE/ISET Conference on Human Vision, Visual Processing, and Digital Display. 2179:310-317.

ATSALAKIS, A., KROUPIS, N., SOUDRIS, D., and PAPAMARKOS, N., 2002a. A Window-Based Color Quantization Technique and its Architecture Implementation. Proceedings of International Conference on Digital Signal Processing, 285-288.

ATSALAKIS, A., PAPAMARKOS, N., and ANDREADIS, I., 2002b. On Estimation of the Number of Image Principal Colors and Color Reduction through Self-Organized Neural Networks. Wiley Periodicals, Inc. Int J Imaging Syst Technol, 12:117-127.

BLOOMBERG, D., 2008a. Color quantization using octrees, Leptonica, http://www.leptonica.org/papers/colorquant.pdf (Access Date 3 Feb. 2012)

BLOOMBERG, D., 2008b. Color quantization using modified median cut. Leptonica, http://www.leptonica.org/papers/mediancut.pdf (Access Date: 3 Feb. 2012)

BRUCKER, P., 1977. On the Complexity of Clustering Problems. Optimizations and Operations Research. Springer, 45-54.

CELEBI, M. E., 2009. Effective Initialization of K-Means for Color Quantization. Proceeding of International Conference ICIP, 1649-1652.

CELEBI, M. E., 2011. Improving the performance of k-means for color quantization. Image and Vision Computing, 29:260–271.

CHAADHA, N., TAN, W. Ch., and MENG T. H. Y., 1994. Color Quantization of Images Based on Human Vision Perception. IEEE International Conference on Acoustics, Speech, and Signal Processing, 5:V/89-V/92.

CLARK, D., January 1996. Color quantization using octrees. Dr. Dobb's Journal, 54-57, 102-104.

CLARK, D., July 1995. The popularity algorithm. Dr. Dobb's Journal, 121-128.

COLORNI, A., DORIGO, M., and MANIEZZO, V., 1991. Distributed optimization by ant colonies. Proceedings of European Conference on Artificial Life, 134-142.

Commission International de L'Eclairage, 1986. Colorimetry, in CIE Pub. 15.2, 2nd ed.

DEKKER, A. H., 1994. Kohonen Neural Networks for Optimal Colour Quantization. Network: Computation in Neural Systems, 5:351-367.

DENG, Y., KENNEY, C., MOORE, M. S., and MANJUNATH, B. S., 1999. Peer Group Filtering and Perceptual Color Image Quantization. IEEE International Symposium on Circuits and Systems, 4: 21-24.

DORIGO, M., 1992. Optimization, Learning and Natural Algorithms. PhD thesis, Politecnico di Milano, Italie.

DUAN, H., LIU, S., and LEI, X., 2008. Air Robot Path Planning Based on Intelligent Water Drops Optimization. International Joint Conference on Neural Networks, 1397-1401.

DUAN, H., LIU, S., and WU, J., 2009. Novel intelligent water drops optimization approach to single UCAV smooth trajectory planning. Aerospace Science and Technology, 13:442-449.

EL-MIHOUB, T., NOLLE, L., SCHAEFER, G., NAKASHIMA, T., and HOPGOOD, A., 2006. A Self-Adaptive Hybrid Genetic Algorithm for Color Clustering. Proceedings of International Conference on Systems, Man, and Cybernetics, 3158-3163.

FLOYD, R. W., and STEINBERG, L., 1976. An adaptive algorithm for spatial grey scale. Proceedings of the Society of Information Display 17:75-77 (1976).

FRANTI, P., 2000. Genetic algorithm with deterministic crossover for vector quantization. Pattern Recognition Letters, 21:61-68.

FREISLEBEN, B., and SCHRADER, A., 1997a. Color Quantization with a Hybrid Genetic Algorithm. Proceedings of International Conference on Image Processing and Its Applications, 1:86-90.

FREISLEBEN, B., and SCHRADER, A., 1997b. An Evolutionary Approach to Color Image Quantization. Proceedings of International Conference on Evolutionary Computation, 459-464.

GERVAUTZ, M., and PURGATHOFER, W., 1990. A simple method for color quantization: octree quantization. Glassner, A., ed, Graphics Gems I, Acad. Press, 287-293.

GONZALEZ, A. I., GRANA, M., ALBIZURI, F. X., D'ANJOU, A., and TORREALDEA, F. J., 2000. A near real-time Evolution-based Adaptation Strategy for dynamic Color Quantization of image sequences. Information Sciences, 122:161-183.

HADDAD, R.A., and AKANSU, A.N., 1991. A Class of Fast Gaussian Binomial Filters for Speech and Image Processing. IEEE Transactions on Acoustics, Speech and Signal Processing, 39:723-727.

HECKBERT, P. S., 1982. Color Image Quantization for Frame Buffer Display. Proceedings of ACM SIGGRAPH, 16(3):297-307.

HILL, F. S. Jr., 1990. Computer Graphics. Macmillan Publishing Co.

HOLLAND, J. H., 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, Michigan.

HORNG, M. H., and JIANG, T. W., 2011. Image vector quantization algorithm via honey bee mating optimization. Expert Systems with Applications, 38:1382-1392.

KAMKAR, I., AKBARZADEH, M. R., and YAGHOOBI, M., 2010. Intelligent Water Drops a new optimization algorithm for solving the Vehicle Routing Problem. Proceedings of International Conference on Systems, Man, and Cybernetics, 4142-4146.

KANDEL, E. R., SCHWARTZ, J. H., and JESSEL, T. M., 1991. Principles of Neural Science. 3rd ed, New York, Appleton and Lange.

KASUGA, H., YAMAMOTO, H., and OKAMOTO, M., 2000. Color Quantization Using the Fast K-Means Algorithm. Systems and Computers in Japan, 31(8):33-40.

KAVEH, A., and TALATAHARI, S., 2010. A Novel Heuristic Optimization Method: Charged System Search. Acta Mechanica 213(3-4):267-289.

KENNEDY, J., and EBERHART, R., 1995. Particle Swarm Optimization. Proceedings of International Conference on Neural Networks, 4: 1942-1948.

KOLPATZIK, B. W., and BOUMAN, C. A., 1992. Optimized error diffusion for image display. Journal of Electron. Imag., 1:277-292.

KRISHNANAND, K. N., and GHOSE, D., 2005. Detection of multiple source locations using a glowworm metaphor with applications to collective robotics. IEEE Swarm Intelligence Symposium, 84-91.

KRUGER, A., September 1994. Median-cut color quantization. Dr. Dobb's Journal, 46-54, 91-92.

KWAK, N. J., RYU, S. P., KWON, H. B., AHN, J. H., 2005. The Improved Binary Tree Vector Quantization Using Spatial Sensitivity of HVS. Key Engineering Materials, 277-279:254-258.

Lee DANIEL, C., PAUL, B., and MIKE, M., 1991. Digital Halftoning. Computer Lab and Reference Library, http://www.efg2.com/Lab/Library/ImageProcessing/DHALF.TXT (Access Date 18 Sep. 2012)

LIXIA, Ch., 2009. A New Human Perceptional Color Quantization Algorithm. Third International Conference on Genetic and Evolutionary Computing, 710-713.

MEHRABIAN, R., LUCAS, C., and MOHAGHEGHI, S., 2006. A Novel Numerical Optimization Algorithm Inspired from Weed Colonization. Ecological Informatics, 1(4):355-366.

MOJSILOVIC, A., 2001. Color Quantization and Processing by Fibonacci Lattices. IEEE Transactions on Image Processing, 10(11):1712-1725.

OMRAN, M. G., ENGELBRECHT, A. P., and SALMAN, A., 2005. A Color Image Quantization Algorithm Based on Particle Swarm Optimization. Informatica, 29:261-269.

ÖZDEMIR, D., and AKARUN, L., 2002. A fuzzy algorithm for color quantization of images. Pattern Recognition, 35:1785-1791.

PALUS, H., 1998. Colour spaces. Chapmann and Hall.

PAN, Sh. M., and CHENG, K. Sh., 2007. An evolution-based tabu search approach to codebook design. Pattern Recognition, 40:476-491.

PAPAMARKOS, N., 1999. Color Reduction Using Local Features and a Kohonen Self-Organized Feature Map Neural Network. John Wiley & Sons, Inc. Int J Imaging Syst Technol, 10:404-409.

PAPAMARKOS, N., ATSALAKIS, A. E., and STROUTHOPOULOS, Ch. P., 2002. Adaptive Color Reduction. IEEE Transactions on Systems, Man, and Cybernetics, 32:44-56.

PEDERSEN, M. E. H., 2010a. Tuning & Simplifying Heuristical Optimization. PhD thesis, School of Engineering Sciences, University of Southampton, England.

PEDERSEN, M. E. H., 2010b. Good parameters for differential evolution. Technical Report HL1002 (Hvass Laboratories).

PEDERSEN, M. E. H., and CHIPPERFIELD, A. J., 2008. Parameter tuning versus adaptation: proof of principle study on differential evolution. Hvass Laboratories Technical Report no. HL0802.

PEDERSEN, M. E. H., and CHIPPERFIELD, A. J., 2008. Parameter tuning versus adaptation: proof of principle study on differential evolution. Hvass Laboratories Technical Report no. HL0802.

PEDERSEN, M. E. H., and CHIPPERFIELD, A. J., 2011. Tuning differential evolution for artificial neural networks. In S.J. Kwon, editor, Artificial Neural Networks. Nova Publishers.

PEDERSEN, M., and HARDEBERG, J. Y., 2009. Survey of full-reference image quality metrics. Høgskolen i Gjøviks rapportserie, nr. 5.

PHAM, D. T., GHANBARZADEH, A., KOC, E., OTRI, S., RAHIM, S., and ZAIDI, M., 2005. The Bees Algorithm. Technical Note, Manufacturing Engineering Centre, Cardiff University, UK.

PHAM, D. T., GHANBARZADEH, A., KOC, E., OTRI, S., RAHIM, S., and ZAIDI, M., 2006. The Bees Algorithm – A Novel Tool for Complex Optimisation Problems. Proceedings of IPROMS Conference, 454-461.

POYNTON, Ch., 1995. A Guided Tour of Color Space, New Foundations for Video Technology. Proceedings of the SMTPE Advanced Television and Electronic Imaging Conference, 167-180.

PRICE, K., 1994. Genetic Annealing. Dr. Dobb's Journal, October.

PRICE, K., STORN, R. M., and LAMPINEN, J. A., 2005. Differential Evolution: A Practical Approach to Global Optimization. Springer.

PUZICHA, J., HELD, M., KETTERER, J., BUHMANN, J. M., and FELLNER, D. W., 2000. On Spatial Quantization of Color Images, IEEE Transactions on Image Processing, 9(4):666-682.

RABANAL, P., RODRIGUEZ, I., and RUBIO, F., 2007. Using River Formation Dynamics to Design Heuristic Algorithms. Unconventional Computation, Springer, 163-177.

RASHEDI, E., NEZAMABADI-POUR, H., and SARYAZDI, S., 2009. GSA: a gravitational search algorithm. Information Science, 179(13):2232-2248.

RAYAPUDI, S. R., 2011. An Intelligent Water Drop Algorithm for Solving Economic Load Dispatch Problem. Electrical and Electronics Engineering 5(1):43-49.

SCHAEFER, G., and NOLLE, L., 2006. A hybrid Differential Evolution approach to colour map generation. Proceeding of 20th European Conference on Modeling and Simulation, 434-436.

SCHAEFER, G., and NOLLE, L., 2006. Generic Black Box Optimisation Algorithms for Colour Quantisation. Applications of Soft Computing - Recent Trends. Springer, 15-21.

SCHEUNDERS, P., 1997. A genetic c-means clustering algorithm applied to color image quantization. Pattern Recognition, 30(6):859-866.

SHAH, B., DHATRIC, P., RAGHAVAN, V., 2004. Using inverse image frequency for perception-based color image quantization. 6th IEEE Southwest Symposium on Image Analysis and Interpretation, 71-75.

SHAH-HOSSEINI, H., 2001b. Otsu's Criterion-based Multilevel Thresholding by a Nature-inspired Metaheuristic called Galaxy-based Search Algorithm. Third World Congress on Nature and Biologically Inspired Computing, 383-388.

SHAH-HOSSEINI, H., 2007. Problem Solving by Intelligent Water Drops. IEEE Congress on Evolutionary Computation, 3226-3231.

SHAH-HOSSEINI, H., 2008. Intelligent water drops algorithm - A new optimization method for solving the multiple knapsack problem. Intelligent Computing and Cybernetics, 1(2):193-212.

SHAH-HOSSEINI, H., 2009. The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm. Bio-Inspired Computation, 1(1/2):71-79.

SHAH-HOSSEINI, H., 2011a. Principal components analysis by the galaxy-based search algorithm: a novel metaheuristic for continuous optimization. International Journal of Computational Science and Engineering 6(1/2):132-140.

SHARMA, G., and TRUSSELL, H. J., 1997. Digital color imaging. IEEE Transactions on Image Processing, 6:901-932.

SHI, Y., and EBERHART, R. C., 1998. A modified particle swarm optimizer. Proceedings of International Conference on Evolutionary Computation, 69-73.

STORN, R., 1996. On the usage of differential evolution for function optimization. Biennial Conference of the North American Fuzzy Information Processing Society (NAFIPS), 519-523.

STORN, R., and PRICE, K, 1997. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 11:341-359.

TAŞDIZEN, T., AKARUN, L., and ERSOY, C., 1998. Color quantization with genetic algorithms. Signal Processing: Image Communication, 12:49-57.

TKALCIC, M., TASIC, J. F., 2003. Colour spaces – perceptual, historical and applicational background. University of Ljubljana, Trzaska 25, 1001 Ljubljana, Slovenia.

VAISEY, J., and GERSHO, A., 1988. Simulated Annealing and Codebook Design. Proceedings of International Conference on Acoustics, Speech, and Signal Processing, 1176-1179.

VEREVKA, O., BUCHANAN, J. W., 1995. Local K-means Algorithm for Color Image Quantization. Proceedings of Graphics Interface Conference, 128-135.

WRIGHT, W. D., 1981. 50 years of the 1931 CIE Standard Observer. Die Farbe, 29:4/6.

WYSZECKI, G., STILES, W.S., 2000. Color Science Concepts and Methods, Quantitative Data and Formulae, John Wiley and Sons, Inc.

XINRONG, H., TIANZHEN, W., and DEHUA, L., 2005. A New Approach of Color Image Quantization. Journal of Communication and Computer, 2(3):72-77.

YANG, X. S., 2008. Nature-Inspired Metaheuristic Algorithms. Luniver Press.

YANG, X. S., 2009. Cuckoo search via Levy flights. International Congress on Nature & Biologically Inspired Computing, 210-214.

YANG, X. S., and DEB, S., 2010. Engineering optimisation by cuckoo search. International Journal of Mathematical Modelling and Numerical Optimisation 1(4):330-343.

YOON, K. J., KWEON I. S., 2004. Human Perception Based Color Image Quantization. Proceedings of the 17th International Conference on Pattern Recognition, 1:664-667.

ZHENG, X., JULSTROM, B. A., and CHENG, W., 1997. Design of Vector Quantization Codebooks Using a Genetic Algorithm. Proceedings of International Conference on Evolutionary Computation, 525-529.

ZHOU, Y. M., LI, S. M., and TANG, M., 2008. A BP Neural Network-Based Color Space Quantization Scheme. Proceedings of the Seventh International Conference on Machine Learning and Cybernetics, 2690-2694.

CURRICULUM VITAE

Amir Poorsadeg-Zadeh-Yeganeh, was born in May 1982 in Urmia, West Azerbaijan, Iran. His father and mother were both teachers and they had a deep impression on his education level and educational discipline. He completed his elementary school, guidance school, high school and pre-university studies in his hometown, Urmia (Orumiyeh) with high grades. At high school, he achieved regional third place in young section of the 11th Khwarizmi Awards for making a digital oscilloscope without using Cathode Ray Tube. After finishing the pre-university studies, he passed the entrance examination of the public universities successfully and entered Urmia University, in the field of Solid State Physics. He was interested in physics and during his studies at Urmia University he was honored for his contributions in Science Association of Physics Students. Nonetheless, his increasing interest in Computer Science finally led him to change his major.

After retaking the entrance examination, in Oct. 2006, he entered Urmia Technical College in Computer Software Technology within Associate degree program. He graduated in Aug. 2008 as the top student of Computer Science department. Then he passed Associate-Bachelor entrance exam and entered Urmia University College of Science and Technology in order to get his BSc degree in Computer Applied Science - Software. He graduated in Aug. 2010 again as top student.

In Sep. 2010, he began his MSc studies at Çukurova University (Adana, Turkey) in Computer Engineering. In Apr. 2012, he achieved European ERASMUS scholarship within student exchange program for Augsburg University, Germany. However, for some reasons he couldn't attend in ERASMUS program and he completed his Master studies in Sep. 2012.

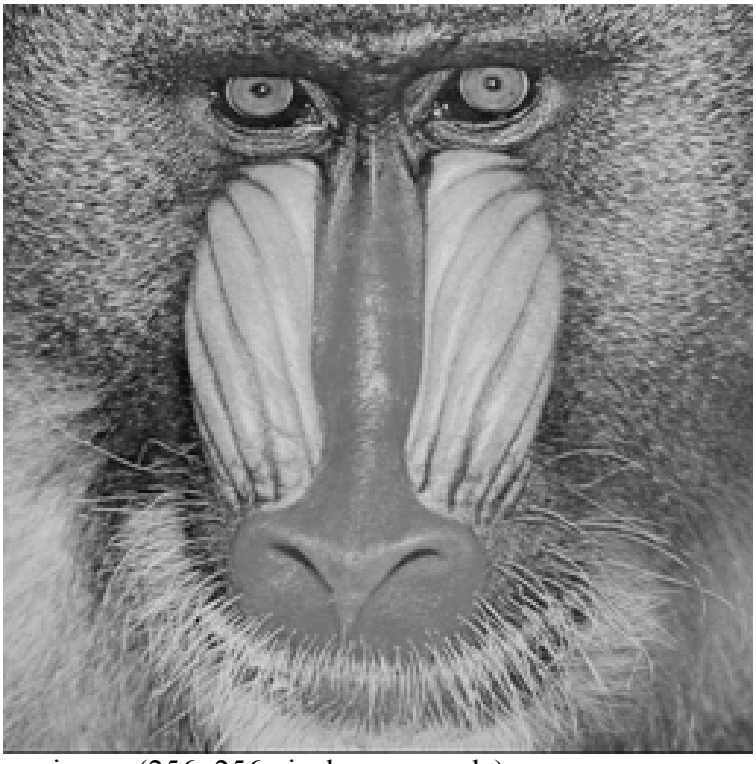
His academic interests are Artificial Intelligence, especially Evolutionary and Nature Inspired Computation, and Image Processing. He is also interested in medical signal processing, programming and Electronic circuits.

APPENDIX

Original Peppers image (256×256 pixels, gray-scale)

Original House image (256×256 pixels, gray-scale)

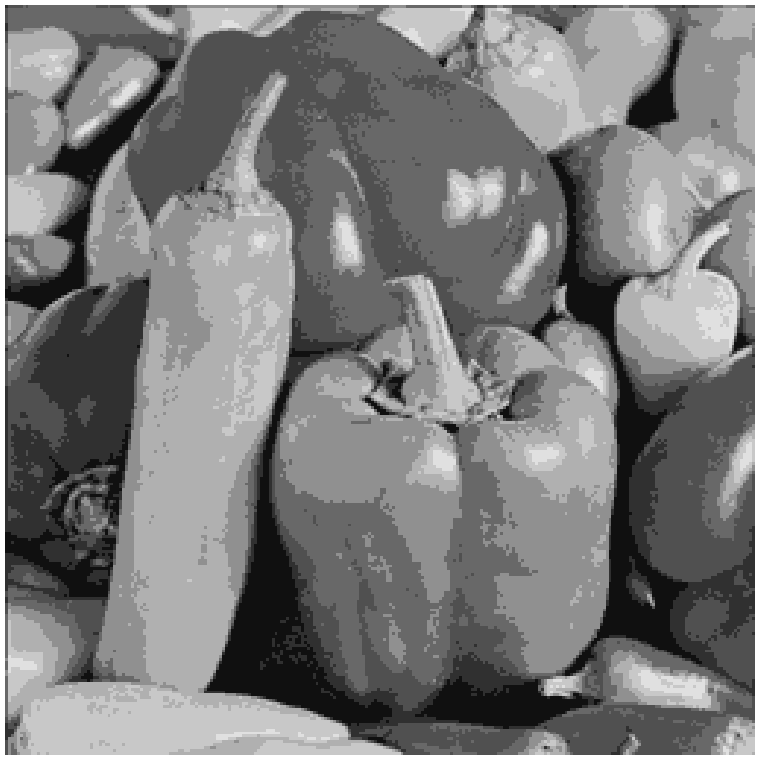
Original Lena image (256×256 pixels, gray-scale)



Original Baboon image (256×256 pixels, gray-scale)

Peppers image quantized to 8 gray levels by P-CQ algorithm (Popularity CQ)

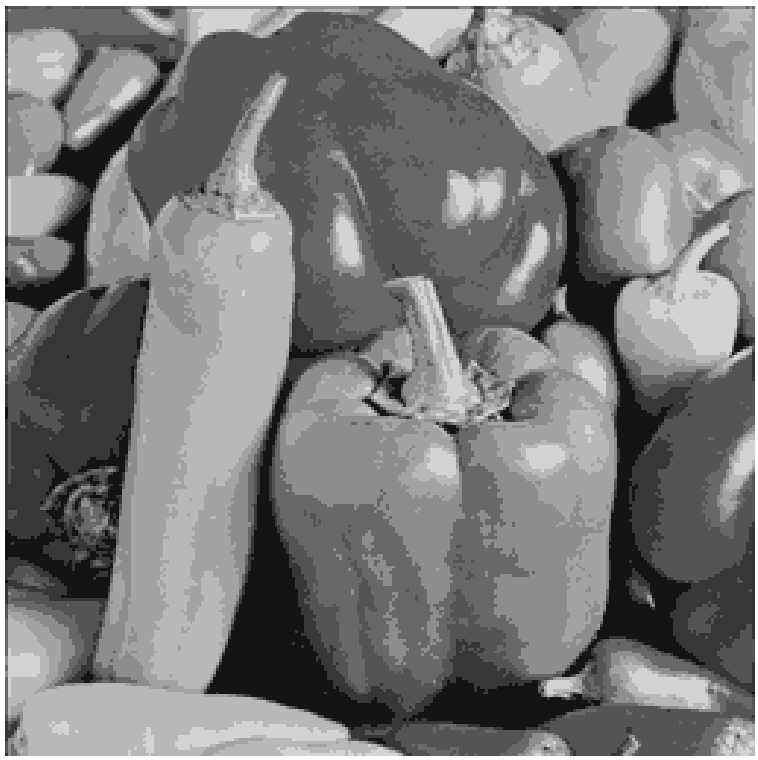
Peppers image quantized to 8 gray levels by U-CQ algorithm (Uniform CQ)



Peppers image quantized to 8 gray levels by O-CQ algorithm (Octree CQ)

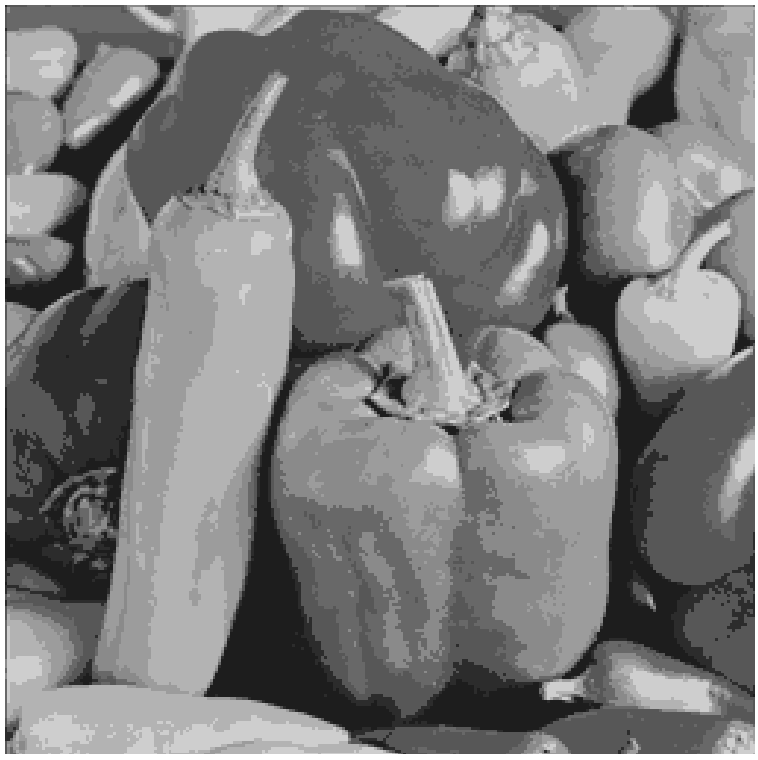
Peppers image quantized to 8 gray levels by MC-CQ algorithm (Median-Cut CQ)

Peppers image quantized to 8 gray levels by Km-CQ algorithm (K-means CQ)



Peppers image quantized to 8 gray levels by SOM algorithm (Self Organizing Maps)

Peppers image quantized to 8 gray levels by ACO algorithm (Ant Colony Optimization)



Peppers image quantized to 8 gray levels by SA algorithm (Simulated Annealing)

Peppers image quantized to 8 gray levels by IWD-CQ algorithm, Con-HUD Heuristic

Peppers image quantized to 8 gray levels by IWD-CQ algorithm, Pop-HUD Heuristic

Peppers image quantized to 8 gray levels by IWD-CQ algorithm, Med-HUD Heuristic

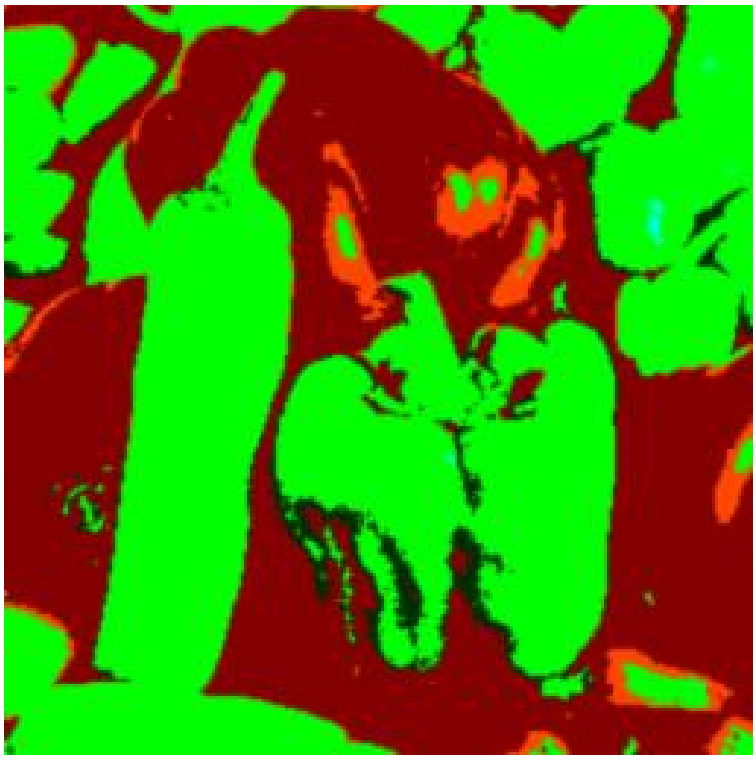
Peppers image quantized to 8 gray levels by IWD-CQ algorithm, P-M-HUD Heuristic

Original Peppers image (256×256 pixels)

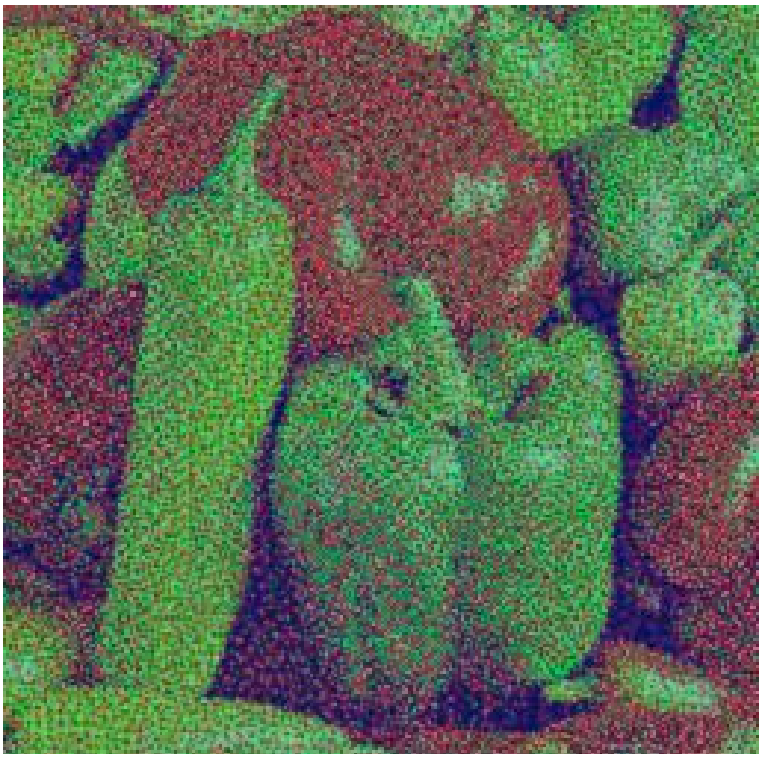
Original House image (256×256 pixels)

Original Lena image (256×256 pixels)

Original Baboon image (256×256 pixels)



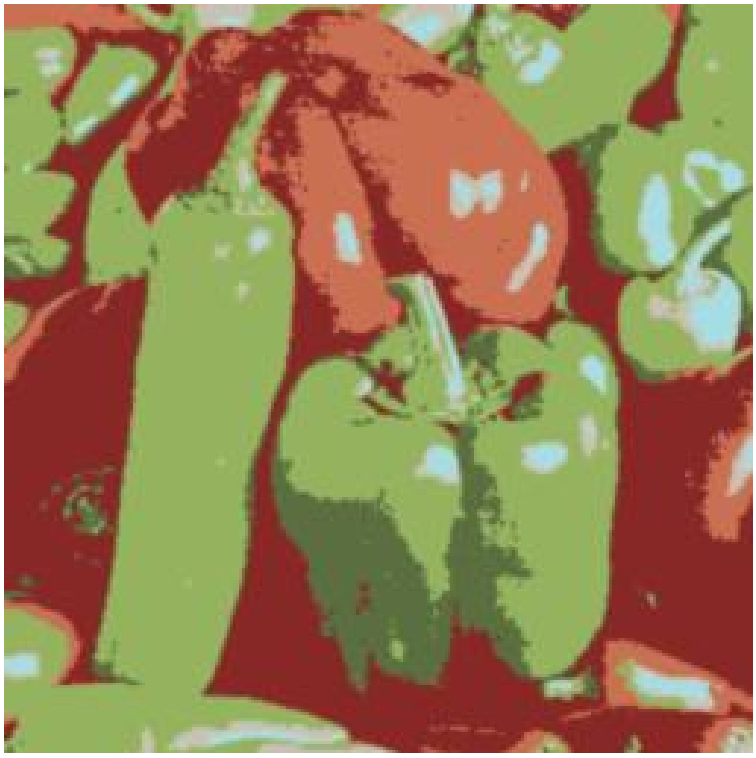
Peppers image quantized to 8 colors by U-CQ algorithm (Uniform CQ) "No Halftoning"



Peppers image quantized to 8 colors by U-CQ algorithm (Uniform CQ) "Halftoning"

Peppers image quantized to 8 colors by P-CQ algorithm (Popularity CQ) "No Halftoning"

Peppers image quantized to 8 colors by P-CQ algorithm (Popularity CQ) "Halftoning"



Peppers image quantized to 8 colors by O-CQ algorithm (Octree CQ) "No Halftoning"

Peppers image quantized to 8 colors by O-CQ algorithm (Octree CQ) "Halftoning"

Peppers image quantized to 8 colors by MC-CQ algorithm (Median-Cut CQ) "No Halftoning"

Peppers image quantized to 8 colors by MC-CQ algorithm (Median-Cut CQ) "Halftoning"

Peppers image quantized to 8 colors by Km-CQ algorithm (K-means CQ) "No Halftoning"

Peppers image quantized to 8 colors by Km-CQ algorithm (K-means CQ) "Halftoning"

Peppers image quantized to 8 colors by DE-PCQ algorithm *objFuncBCQ* "No Halftoning"

Peppers image quantized to 8 colors by DE-PCQ algorithm *objFuncBCQ* "Halftoning"

Peppers image quantized to 8 colors by DE-PCQ algorithm *objFuncPCQ* "No Halftoning"

Peppers image quantized to 8 colors by DE-PCQ algorithm *objFuncPCQ* "Halftoning"