Software Engineering Department
Ort Braude College

Attribution of Paintings

In Partial Fulfillment of the Requirements for

Final Project in Software Engineering (Course 61401)

Karmiel – July 2018

Hadi Abu-Maruf - 301188504

Tomer Barak – 305284275

Supervisor: Dr. Elena Ravve

Contents

Abbreviations

MGGD Multivariate Generalized Gaussian Distribution
GD Geodesic Distance
KLD Kullback Leibler Distance
DWT Discrete Wavelet Transform
MM Method of Moments
ML Maximum Likelihood
PAM Partition Around Medoids

1. INTRODUCTION

1.1. Organization of the Paper

The book organized as follow:

- Section 1 reviews the introduction of the project. What are we going to do? Why is the project not-trivial, and what are the difficulties of this project?
- Section 2 reviews the historical overview, related works.
- Section 3 describes our approach in details-the main stream of the project.
- Sections 4 expected results.
- Sections 5 focuses on the software engineering documentation: use case diagrams, class diagrams, test plan and GUI.

1.2. What are we Going to Do?

In this project, our objective is to invent an algorithm and its implementation, which attributes a given painting to known artists, which can run in a reasonable time.

At first, we get a professional database, which contains a big number of paintings of famous images, like Pablo Picasso, Vincent van Gogh. We decided that the size of each window is be $128X128$. Then we implement wavelet transform in every window. After the Wavelet transform process, we cluster the database by using PAM algorithm. Then we can compare the given painting with the clustered database in order to find the similar paintings that suitable with the given painting based in “Pareto optimal solutions”.

We start with a painting and a database an input and at the end of our process; we get candidates for similar painting that are the most suitable, that mean have the same histogram style and it can give us information about the author.

1.3. Why is the Project not Trivial?

First, until this day there is no solution in computer vision field to sort a painting by painter.

In our project, we decided to try a way that converts the painting to a variance matrix. Then, we expect that the coding of the painting to include different values, related to the painting style. Next, we should find a way to compare between the input paintings with our database.

1.4. Difficulties in Setting this Project

1. The main problem is how to cluster and group the paintings to matrices.

2. It seems that we need multi-criteria approach in definition of the similarity of the elements.
3. Associating the painting to the correct painter.

2. THEORY

2.1. Related Work

This project describes a solution of dating challenge, i.e. definition of painting to a particular period. This challenge is not simple. Technical analyses of the painting, including of pigments present, the materials used and the method of their preparation. Visual inspection of the painting is important as well to explore and characterize the style of the work. The solution of this challenge based on painting processing and uses computational tools from painting analysis and machine learning as painting segmentation wavelet coefficients and Pam clustering to reveal characteristics of painting. This approach allows finding out similarity between paintings from the same period.

In region-based painting retrieval, texture features are most important in determining the class a region belongs to, since they can overcome the limitations of color and shape features. Two reliable approaches to model texture features are wavelet transform. Using these filters allow us to transform texture paintings into frequency domain. Then, such methods can generate sufficient number of features to classify the painting. Although both tools are close to human visual perception, a lot of information has to be extracted from their windows for effective texture classification.

2.2. Background

In this project, in order to achieve the project proposes, we have to realize and act according to the following 4 main steps:

1. Divide the painting to somewhat small rectangles (windows) and calculating a wavelet transform on each.
2. Find similarities between windows based on their wavelet transform coefficient histogram.
3. Use a clustering Pam algorithm for the final texture classification.
4. Build a database that contains the paintings to compare with.

2.3. The Main Stream of the Project

We divide the process into the following super-steps:

Database – preprocessing:

1. Initial search for paintings and DB creation
2. Converting of paintings to RGB format.
3. Division of the painting to windows
4. Wavelet transform algorithm on each window.
5. Parameters estimations for calculating variance in each window.
6. Sum the three RGB variance matrices into one matrix.
7. Calculating Geodesic distance.
8. PAM clustering.

New painting attribution:

1. Association a new painting to a cluster.
2. Association a new painting to its painter.

2.3.1. Database Pre-proceeding

1. Initial paintings and DB creation

- Input of the step

Number of paintings for each potential artist, in order to get as much information and texture characteristics as possible, to obtain the most accurate result. At this stage, we also classify all paintings in two vectors - each vector contains paintings of a specific artist.

- What is the local problem to be solved?

The difficulty of this stage is it requires all paintings to be in high resolution, minimum 512x512.

- What is the local result that we want to reach?

We want to have only paintings with 512x512 resolution or higher.

- Why do we need the result?

This condition guarantees sufficient details of the paintings.

- How do we reach the result?

We exclude paintings with resolution lower than 512x512.

- Output of the step

We get as output two vectors of paintings, and a test painting. Each painting is a collection of pixels in resolution at least 512x512.

2. Converting of paintings to RGB format

- Input of the step

At this stage, we get as input a matrix of pixels for each painting from the previous stage, and convert them to RGB format. At Fig. 2, we can see example of a painting to RGB format. On the left, we see the original painting. On the right (from up to down) the original painting in Red, Green and Blue respectively.

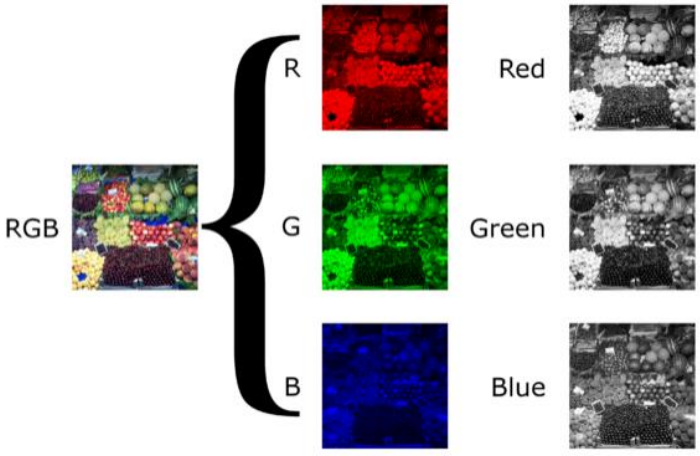


Fig. 2 - RGB format

- What is the local problem to be solved?

Separation of color paintings to three basic colors: Red, Green, Blue.

- What is the local result that we want to reach?

RGB representation for each painting.

- Why do we need the result?

As known, texture of painting is determined by combination of colors. Each color actually is a combination of three basic colors: Red, Green and Blue. In contrast to the human eye, computer perceives the color as numeric values in range from 0 to 255 for each color, where 0 is minimal intensity of color, and 255 its maximum value.

- How do we reach the result?

Every painting is represented as 3D matrix, where each layer of the matrix is a two- dimensional matrix of red, green or blue pixel values. We then get (:, :, X), where X may be 1, 2 or 3 according to the colors. For example, if we need to get red component of "painting", we use Image (:, :, 1) at Matlab.

- Output of the step

As output we get all input paintings in RGB format, represented by matrices.

3. Division of the Painting to windows for each painting

- Input of the step

Image in RGB format.

- What is the local problem to be solved?

Divide each painting to windows; each windows size is 128*128

- What is the local result that we want to reach?

Divide painting to windows.

- Why do we need the result?

This step helps us to find similarity between windows in each painting.

- How do we reach the result?

Run our painting division algorithm.

- Output of the step

We get as output collection of matrices, which represent windows.

4. Wavelet transform algorithm

- Input of the step

From the previous stage we run in each window wavelet algorithm (see Fig. 2).

- What is the local problem to be solved?

Run wavelet transform algorithm.

- What is the local result that we want to reach?

Details about the window that give us information about the window in 4 aspects, C_H the coefficient for the horizontal decomposition, C_V the vertical decomposition, C_D the coefficient for the diagonal decomposition and C_A.

- Why do we need the result?

To get DWT coefficient about the window.

- How do we reach the result?

Running our wavelet transform algorithm

$$CA(x,y) = (W(i,j) + B(i,j+1) + W(i+1,j) + W(i+1,j+1))/2;$$

$$CH(x,y) = (W(i,j) + W(i,j+1) - W(i+1,j) - W(i+1,j+1))/2;$$

$$CV(x,y) = (W(i,j) - W(i,j+1) + W(i+1,j) - W(i+1,j+1))/2;$$

$$CD(x,y) = (W(i,j) - W(i,j+1) - W(i+1,j) + W(i+1,j+1))/2;$$

CA is not important in our steps, so for our steps we just need CH, CV, CD each one is matrix of $128 * 128$

$x, y = 1, \dots, 128$

- Output of the step

$\sum_i = C_H, C_V, C_D$ matrices, each on is $128 * 128$

5. Parameters estimations for calculating variance in each window

- Input of the step

$\sum_i = C_H, C_V, C_D$ matrices from the previous step.

- What is the local problem to be solved?

We need to compute variance.

What is the local result that we want to reach?

We want to compute variance for CH, CV, CD.

-Why do we need the result?

$\sum_i = var_H, var_V, var_D$ to get details about each window.

- How do we reach the result?

By calculating, the variance of each window (see section 4.2.1 Parameters estimations).

- Output of the step

$$Var = \begin{pmatrix} \sum_{H1} & \sum_{V1} & \sum_{D1} \\ \sum_{H2} & \sum_{V2} & \sum_{D2} \\ \vdots & \vdots & \vdots \\ \sum_{Hn} & \sum_{Vn} & \sum_{Dn} \end{pmatrix}$$

6. Sum the three RGB variance matrices into only one matrix

- Input of the step

From the previous stage, we get Var matrix.

- What is the local problem to be solved?

Sum three matrices into one matrix.

- What is the local result that we want to reach?

Matrix of Var_RGB, using a simple sum of each three cells in each i, j.

- Why do we need the result?

Checking the similarity between windows has to be test on sum of three RGB matrices of the picture. We need to work with one matrix instead of three matrices.

- How do we reach the result?

By summing each i, j with k, r in other two matrices, while $i = k, j = r$.

- Output of the step

$$Var\_RGB = \begin{pmatrix} \sum_{H1} & \sum_{V1} & \sum_{D1} \\ \sum_{H2} & \sum_{V2} & \sum_{D2} \\ \vdots & \vdots & \vdots \\ \sum_{Hn} & \sum_{Vn} & \sum_{Dn} \end{pmatrix}$$

7. Calculating geodesic distance

- Input of the step

From the previous stage we get Var_RGB matrix, $\beta = 0.5$, mm=1.

- What is the local problem to be solved?

Calculating two points on the MGGD manifold is the shortest curve to connect between two MGGDs $GD(\beta, \sum_1), (\beta, \sum_2)$ points.

- What is the local result that we want to reach?

Matrix of $\sum$ each row represents similarity between $\sum_i$ to $\sum_j$, $i \neq j$ in different paintings.

- Why do we need the result?

Matrix of distance between windows that gives us information about the similarity between the windows.

- How do we reach the result?

We use the method geodesic distance.

- Output of the step

Matrix of n*n that represent distance between each two windows in different paintings.

$$D = \begin{pmatrix} d_{1,1} & d_{1,2} & \cdots & d_{1,n} \\ d_{2,1} & d_{2,2} & \cdots & d_{2,n} \\ \vdots & \vdots & & \vdots \\ d_{n*k,1} & d_{n,2} & \cdots & d_{n*k,n} \end{pmatrix}$$

8. PAM (Partitioning around Medoids) clustering

- Input of the step

From the previous stage, we get D matrix.

- What is the local problem to be solved?

In the worst case, a painting can be belonged to two potential styles, because the distance is equal to both of their medoids, hence the decision is random.

- What is the local result that we want to reach?

Finding the best k-medoids result provides the most accurate clustering.

- Why do we need the result?

Clustering painting's style provides more details for discovering its artist.

- How do we reach the result?

Running k-medoids algorithm (see section 3.5).

- Output of the step

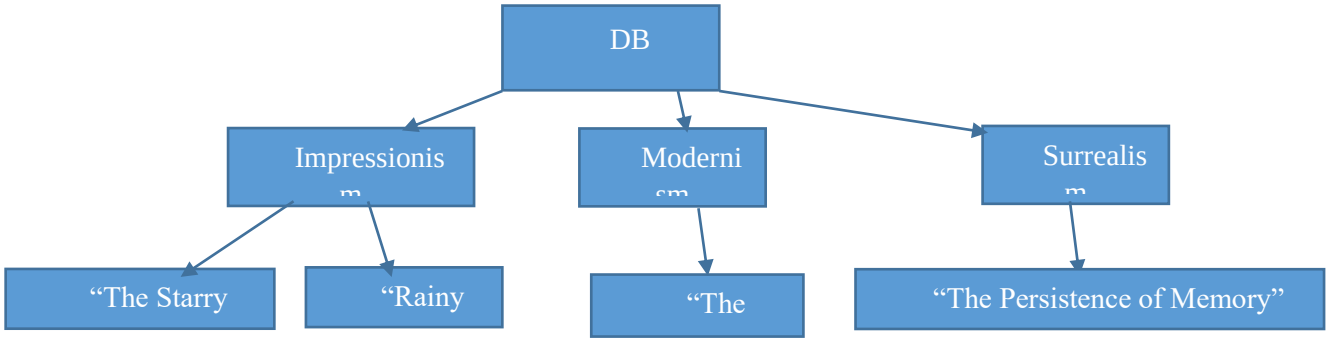


Fig. 3 - DB clustering example

2.3.2. Super-steps of the Input Painting Proceeding

In addition, the database steps from 1-8 that we have to implement. In the new painting, we have to implement the following steps:

1. Associate a new painting to a cluster

- Input of the step

We implement steps 1-7 from the pre-processing steps and get:

$Input\_D = \begin{pmatrix} 0 & d_{1,2} & \cdots & d_{1,36} \\ d_{2,1} & 0 & \cdots & d_{2,36} \\ \vdots & \vdots & & \vdots \\ d_{36,1} & d_{n,2} & \cdots & 0 \end{pmatrix}$ matrix represents GD between each two windows of input and the clustered database.

- What is the local problem to be solved?

We need to associate the painting to the suitable cluster.

- What is the local result that we want to reach?

Associating a painting to its suitable cluster.

- Why do we need the result?

Painting's right cluster provides us information that helps to know to whom painter the painting belongs.

- How do we reach the result?

We take our $Input\_D$ matrix and calculate the minimal Euclidian Distance from each medoid in each cluster.

- Output of the step

Input painting is associated to a cluster.

2. Associate a new painting to its painter

- Input of the step

$D\_Vec = (d_1, d_2, \dots, d_k)$ vector that represents the distances between input and all k-clusters.

- What is the local problem to be solved?

To find the most suitable paintings candidates from the cluster to the input painting.

- What is the local result that we want to reach?

N (say 5) most suitable candidates to the input painting.

- Why do we need the result?

We want to associate the input painting to the author, and from the candidates painting, we can learn to whom author the painting belongs.

- How do we reach the result?

In this final step, we only take in account the cluster with a minimal distance. Then we have to recognize which painter's paintings is the most similar to our input painting. We have a specific method (The Pareto Algorithm) that we need in order to find which paintings are the closets to ours.

- Output of the step

Present the most similar paintings to the input painting and a potential painter.

2.4. Wavelet Transform

Here, we follow the previous project description [1] . Wavelet is a function that represents a small wave, "mini waves": a short burst of wave that quickly fade away. It oscillates its amplitude above and below zero X value starting and ending at zero X.

The basic function includes different scale and amplitude as follows:

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right) \qquad (1)$$

Here, a is a positive and defines the scale and b is any real number and defines the shift.

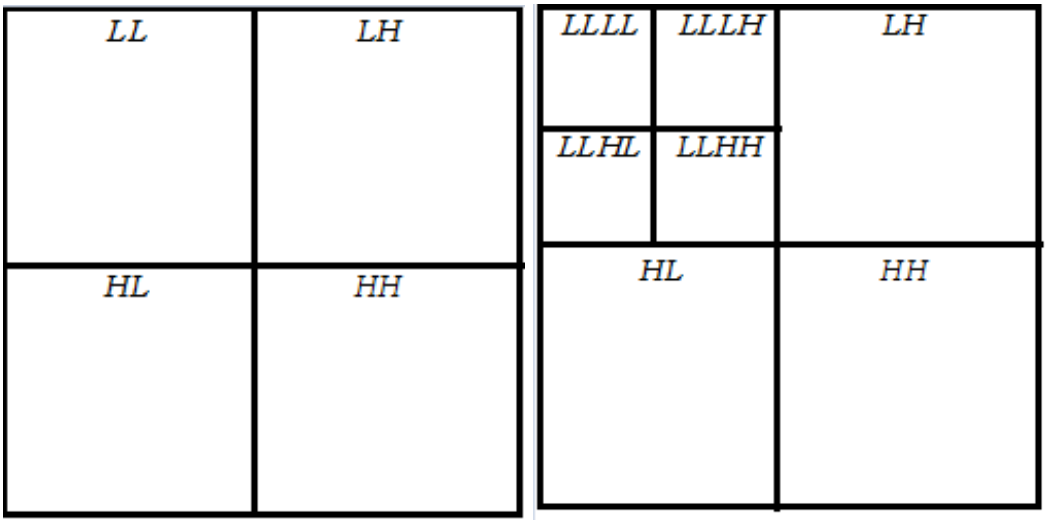
2.4.1. Continuous Wavelet Transform

The wavelet decomposition of $f(x)$ is achieved by the following equation:

$$(f(x), \psi 2^s, t(x)) = \int_{-\infty}^{\infty} f(x)\psi 2^s, t(x)dx \quad (2)$$

Where $\psi 2s,t(x)$ is a family of basic functions obtained by translation and dilation of the mother wavelet function $\psi(x)$.

The wavelet mother function can be constructed either from the multiresolution analysis framework unsupervised texture segmentation or from filter banks.



(a) first scale decomposition (b) second scale decomposition

Fig. 4 – Discrete Wavelet Decomposition

2.4.2. Discrete Wavelet Transform (DWT)

The original painting is first passed through the low-pass and high-pass filters to construct the low-low (LL), low-high (LH), high-low (HL) and the high-high (HH) sub images. Then the decomposition is repeated on the LL sub image to obtain the next 4 sub images (see Fig. 4); and so on until a requested decomposition level is reached.

2.5. Multivariate Generalized Gaussian Distribution

The general Gaussian distribution with zero mean is defined as follows:

$$f(x|\alpha,\beta) = \frac{\beta}{2\alpha\,\Gamma[\frac{1}{\beta}]}\exp[-(\frac{|x|}{\alpha})^{\beta} \quad (3)$$

where Γ is the Gamma function. α obviously is the scale parameter controlling the width of the probability density function [PDF], and $\beta > 1$ is the shape parameter controlling the fall off rate, from Laplacian ($\beta = 1$) to Gaussian ($\beta \geq 2$) distribution. Based on previous study on the multivariate case in the multivariate exponential power distribution probability density function is defined as follows:

$$f(x|\sum,\beta) = \frac{\Gamma(\frac{m}{2})}{\pi^{\frac{m}{2}}\,\Gamma\left(\frac{m}{2\beta}\right)2^{\frac{m}{2\beta}}} * \frac{\beta}{|\sum|^{\frac{1}{2}}}\exp\{-\frac{1}{2}[x'\sum^{-}1x]^{\beta}\} \quad (4)$$

m is the probability space dimension i.e. $m = 3$ for 3 color bands RGB. Σ is the dispersion matrix, equal to the distribution covariance in the Gaussian case. As mentioned β is the shape parameter. X are sample

vectors of the wavelet co- efficient for the image sub regions. $\beta = 1$ yields a Laplacian PDF and $\beta = 2$ a Gaussian one. In this work, we rely on the results from [10] showing the heavy tailed Laplacian PDF outperforms the Gaussian PDF in case of texture discrimination. Hence, in our case the scale parameter β have a fixed number of 1/2.

2.6. Parameters Estimations

For the distance measurement, an estimation of the parameters Σ is required. A comparison between three parameter estimation methods was done for the estimation of the mean, the dispersion and the shape parameter. The method of moments [MM], the maximum likelihood method [ML] and a moment method followed by an optimization through a single Fisher scoring step [Fisher]. Further- more, a goodness of fit test of the multivariate generalize Gaussian distribution for each estimation algorithm has been done on a selected data set. Finally, as we can see from the results presented from that GOF test the ML performed better in the mid-range of the scale parameter β values. So a first estimation of Σ is calculated with MM. And the rest of the work focused on using ML as the parameter estimation algorithm. As a first approximation, we use the MM to estimate the variance of vectors X:

$$\hat{Var} = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^T \qquad (5)$$

Then we estimation for Σ can be calculated:

$$\sum = \hat{Var}(x)\frac{\pi^{\frac{m}{2}}\Gamma\left(\frac{m}{2}\right)2^{\frac{m}{2\beta}}}{\Gamma\left(\frac{m}{2}\right)} \qquad (6)$$

Then we use the ML for optimizing the estimation of Σ as follows:

$$\sum = \frac{\beta}{n}\sum_{i=1}^{n} u_i^{\beta-1} x_i x_i^T \qquad (7)$$

And with $u_i \equiv x_i^T \sum^{-} 1 x_i$ the optimization for $\sum$ can be achieved by recursively solving those equations.

2.7. Similarity Measurement

A lot of work considered the Kullback Leibler divergence (KLD) in order to compute similarity measure between two populations. As we can see in [1] a comparison between two similarity measurements was tested. The KLD and the geodesic distance (GD). In which the GD found to be superior in a few aspects. Hence, we mainly focus on GD.

2.7.1. Geodesic Distance

Geodesic path between two points on the MGGD manifold is the shortest curve to connect the two points that lies within the manifold. The geodesic distance in the case of fixed shape parameter between two MGGDs $GD(\beta, \sum_1), (\beta, \sum_2)$ denotes in a closed form as follows:

$$(\beta 1, \sum_1), (\beta 2, \sum_2) = \left(\left(3b_n - \frac{1}{4}\right)\sum_i \left(r_i^2\right)\left(r_i^2\right) + 2\left(b_n - \frac{1}{4}\right)\sum_{i<j} r_i^2 r_j^2\right)^{\frac{1}{2}} \qquad (8)$$

With $r_i^2 = ln\lambda_2^i$ and i=1, …, m, the m eigenvalues of $\sum_1^{-1}\sum_2$. In addition, b_n is defined as follows:

$$b_n = \frac{1}{4}\frac{m+1}{m+2} \qquad (9)$$

2.7.2. Distances Evaluation

The information distance $d(M,P)$ between two windows M and P as described in this paper can be calculated in different ways. Previous published work has been using two common methods, the GD and the KLD, some even researched the differences between the two methods in terms of performance, accuracy and implementation. Let define the distance $d(M,P)$ as GD between M and P. For a window M in an image let define NM to be a group containing all the nearest surrounding windows (neighbor windows) of M as follows:

$$N_M = \{M_1, M_2, M_3, M_4, M_6, M_7, M_8, M_9\} \quad (10)$$

note that M_5 is missing. M_5 is the subject window M. In that case the distance between a window and its neighbor windows define as follows:

$$d(M, N_p) = \hat{d}(\{d(M, P_i)\}, \forall P_i \in N_p \quad (11)$$

where d is a main calculation. In real cases not, all of a window neighbors are in the same texture. We experiment with different means to minimize the effect of neighbors like that. Then in our first approach the distance between two windows A and B is represented by the following equation:

$$d(A,B) = |d(A,N_A) + d(B,N_B) - (d(A,N_B) + d(B,N_A))| \quad (12)$$

$d(A,N_A)$ is the distance between A and its neighbors, $d(B,N_B)$is the distance between B and its neighbors. Both should sum up to zero in the case A and B matches. $d(A,N_B)$ and $d(B,N_A)$ are the crossed distances between A and B's neighbors and between B and A's neighbors, that should also sum up to zero in the case A and B matches. In all other cases the result should be anything but not zero.

An ϵ value is chosen as a threshold to determine what distances values satisfies similarity between windows and all neighbor windows yielding greater distance value then ϵ is neglected from the average distance calculation. In a formal way.

$$P_i = N_p \Leftrightarrow d(M,P_i) < \epsilon \quad (13)$$

2.8. Clustering

In this project, we use k-medoids clustering, based on PAM [7] clustering as described:

Input, k number of clusters D: the data set containing n items.

Output, a set of k clusters that minimizes the sum of the dissimilarities of all the objects to their nearest medoids.

$$Z = \sum_{i=1}^{k} |x - m_i| \quad (10)$$

Z: Sum of absolute error for all items in the data set.

x: the data point in the space representing a data item.

m_i: is the medoid of cluster c_i

Steps of k-medoids:

Arbitrarily choose k data items as the initial medoids.

Assign each remaining data item to a cluster with the nearest medoid.

Randomly select a non-medoid data item and compute the total cost of swapping old medoid data item with the currently selected non-medoid data item.

If the total cost of swapping is less than zero, then perform the swap operation to generate the new set of k-medoids.

Repeat steps 2, 3 and 4 till the medoids stabilize their locations.

In Fig. 5 with k=4 we can see first loop in the left, and in the right the final loop.

The arrows show that in the beginning we chosen bad medoids, in the middle the we got better results, and in the final loop we can see that the medoids points is close to its clusters and the process ended.

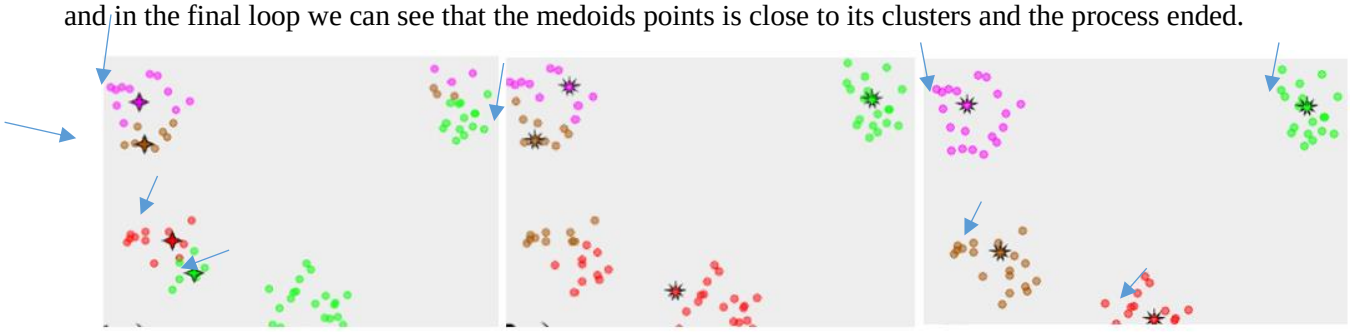


Fig. 5 – PAM example

2.9. Extract the Most Suitable Paintings from the Closest Cluster

At this step we are going to recognize which paintings are the most similar to our input painting; to do this we have a specific approach (The Pareto Algorithm) that we need in order to find which paintings are the most closets to the input painting.

In order to implement this approach, we use multi-criteria optimization, while we work with $N * 3$ dimensions: each $\sum_i$ represents window variance element; C_H, C_V, C_D. Then, we calculate the $N * 3$ distances between the input painting and each painting in the closest cluster.

We check matrices of which paintings have the smallest difference. Paintings of those matrices are the closest paintings of the cluster to the input painting.

2.9.1. Pareto Optimal Solutions

The Pareto optimal solutions are all the solutions that cannot be improved in any of the objectives without degrading at least one of the other objectives.

A feasible solution $x_1 \in X$ is said to Pareto (dominate) another solution $x_2 \in X$, if:

$f_i(x_1) \leq f_i(x_2)$ for all indices $i \in \{1,2, \dots, k\}$.

$f_j(x_1) < f_j(x_2)$ for at least one index $j \in \{1,2, \dots, k\}$.

A solution $x_1 \in X$ is called Pareto optimal, if it is dominated by no other feasible solution, which means that there exists no other solution that is superior at least in case of one objective function value and equal or superior with respect to the other objective functions values. And the function $f(x_1)$ is called Pareto optimal outcomes.

In case of more than one objective function, the Pareto- optimal solutions are rather a class of solutions, forming a surface in objective function space, than a single solution. This surface is commonly called as a Pareto-front.

Therefore, the set of Pareto optimal outcomes $\{f(\mathrm{x})|x \in X, x\ is\ \text{Pareto optimal}\}$ is called the Pareto front or Pareto boundary.

General example:

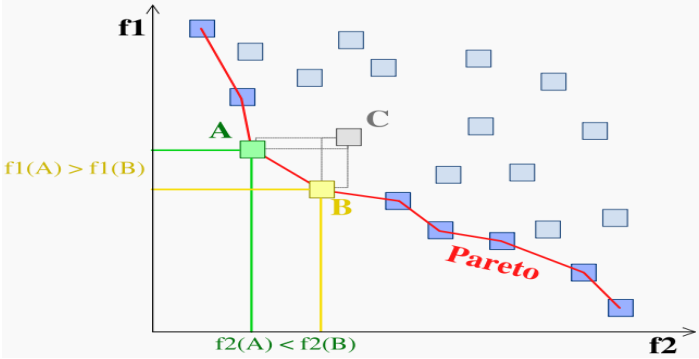


Fig. 6 – Example of the Pareto Curve

For example, consider the problem of optimizing two objective functions: f1(x), f2(x)

The boxed points represent feasible choices, and smaller values are preferred to larger ones because we are talking about min optimization problem.

Point C is not on the Pareto Front. It means that there is another point with a smaller value of the two objective functions. In other words, C is dominated by that point.

In our example, C is dominated by both point A and point B. But points A and B are not strictly dominated by any other, and hence do lie on the frontier.

How do we apply the general approach to our particular problem?

From section 3.2, we have $Input\_\widetilde{Var} = \begin{pmatrix} \widetilde{\sum}_{H1} & \widetilde{\sum}_{V1} & \widetilde{\sum}_{D1} \\ \widetilde{\sum}_{H2} & \widetilde{\sum}_{V2} & \widetilde{\sum}_{D2} \\ \vdots & \vdots & \vdots \\ \widetilde{\sum}_{Hn} & \widetilde{\sum}_{Vn} & \widetilde{\sum}_{Dn} \end{pmatrix}$, the matrix of our input painting, each $\sum_i$ represents a window.

Suppose that the group SET= $\{A\_Var, B\_Var, C\_Var\ , D\_Var....\}$ is a group of matrices that represent different paintings in the same cluster. Each dimension matrix is $3 * N$ as the input matrix. For example, C_Var matrix represented as:

$$C\_Var = \begin{pmatrix} \widetilde{\sum}_{H1} & \widetilde{\sum}_{V1} & \widetilde{\sum}_{D1} \\ \widetilde{\sum}_{H2} & \widetilde{\sum}_{V2} & \widetilde{\sum}_{D2} \\ \vdots & \vdots & \vdots \\ \widetilde{\sum}_{Hn} & \widetilde{\sum}_{Vn} & \widetilde{\sum}_{Dn} \end{pmatrix}$$

Now we are going to implement the Pareto algorithm, in our case:

$$\begin{cases} \min\left(f_1(x), f_2(x), \dots, f_k(x)\right) \\ s.t.\, x \in X \end{cases}, k = 3 * \text{N dimensions}, X \text{ is the feasible set of decision matrix}$$

First of all, we'll define the $f_i(x)$ functions, in our case $f_i(x)$ = The absolute value of the difference $\sum_i$ from the input matrix to $\sum_i i$ in each matrix in the cluster, where $1 \le i,j \le n$

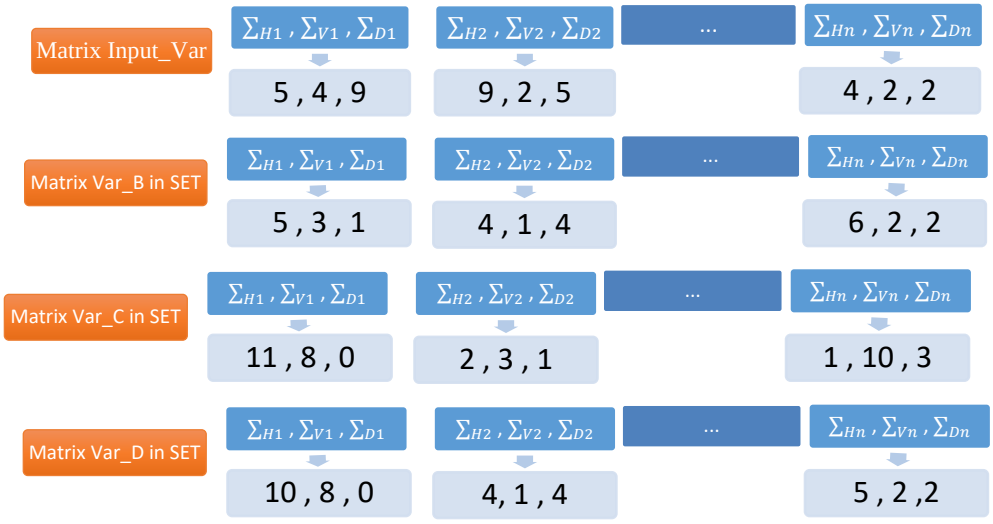


Fig. 7 – Example of the Pareto functions

In the example we will calculate the objective functions values according to the definition between Matrix $Input\_Var$ to B_Var Matrix.

$$\begin{cases} f1(B\_Var) = (|5-5|, |4-3|, |9-1|) = (0,1,8) \\ f2(B\_Var) = (|9-4|, |2-1|, |5-4|) = (5,1,1) \\ \vdots \\ fn(B\_Var) = (|4-6|, |2-2|, |2-2|) = (2,0,0) \end{cases}$$

In the same way, we calculate the objective matrix for the entire matrix in our SET. Then we got a set of the objective matrices $F\_SET = \{ f(A\_Var), f(B\_Var), f(C\_Var), f(D\_Var) \dots . \}$.

As we said before, we pay attention to the Pareto optimal solutions only, so we have to drop out and eliminate matrices that are not Pareto optimal solutions from the F_SET.

In this example, we know that the feasible set includes matrices A, B, C, D.

A and B are the same matrices from the previous example.

The $F\_SET = \{f(A\_Var), f(B\_Var), f(C\_Var), f(D\_Var) \dots . \}$.

We can notice that matrix B_Var dominate matrix C_Var because:

$$\begin{cases} f1(B\_Var) = (0,1,8) < f1(C\_Var) = (6,4,9) \\ f2(B\_Var) = (5,1,1) \le f2(C\_Var) = (7,1,4) \\ \vdots \\ fn(B\_Var) = (2,0,0) < f3(C\_Var) = (3,8,1) \end{cases}$$

C_Var is not a Pareto optimal solution, then we delete it from our SET.

B_Var and D_Var are Pareto optimal solution, because they are not dominated by each other:

$$\begin{cases} f1(B\_Var) = (0,1,8) < f1(D\_Var) = (5,4,9) \\ f2(B\_Var) = (5,1,1) = f1(D\_Var) = (5,1,1) \\ \vdots \\ but\ fn(B\_Var) = (2,0,0) > fn(D\_Var) = (1,0,0) \end{cases}$$

According to the Pareto algorithm, the new set $P.F\_SET = \{B, D\}$ is called Pareto front set.

Now, after we got the Pareto Front Set (that can be quite large) we select five matrices (solutions) by any suitable algorithm, for example, the tail-removing algorithm.

Tail removing:

If we receive many Pareto optimal solutions then we should remove the "tails".

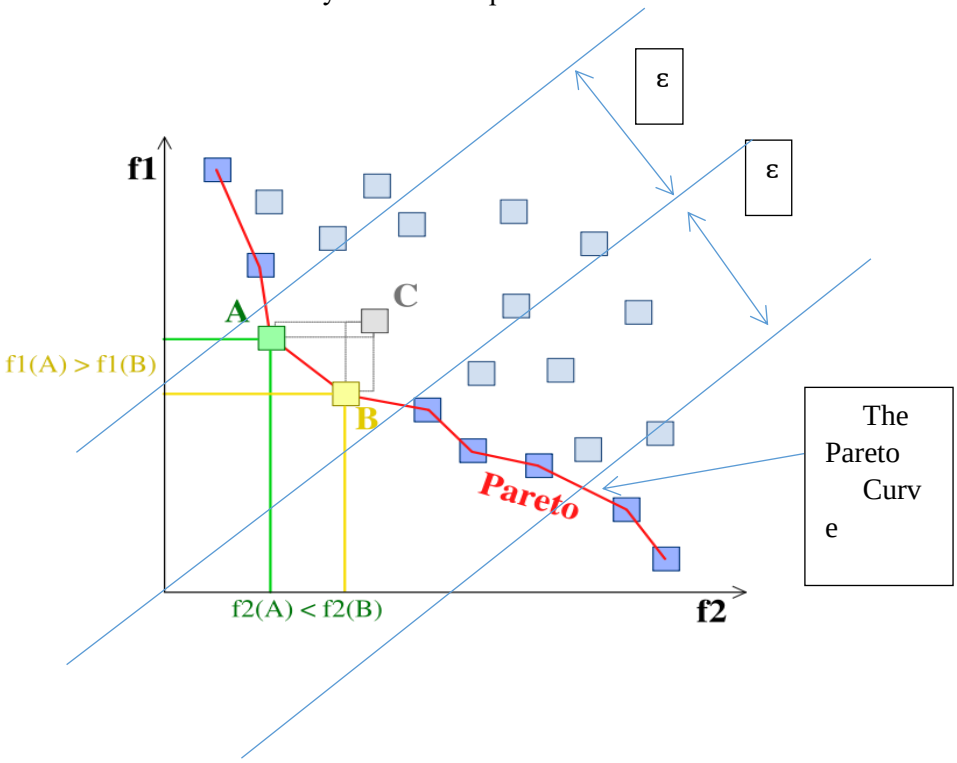


Fig. 8 – Choosing solutions from the Pareto curve

We "draw" straight line representing the identity function. In our particular two-dimensional example with two objective functions, the line is the f1=f2 function.

We need to move away from the line in both directions. We move away by ε in parallel line until we accumulate 5 points in the (+ε, -ε) area and these will be our solution.

2.10. The Overall Method

We are going to start by going through the steps as following; First step is dividing the painting to smaller regions (windows). Then the use of discrete wavelet transforms (DWT) to compute the DWT coefficient for that window. That step gives us: C_H the coefficient for the horizontal decomposition, C_V the coefficient for the vertical decomposition, C_D the coefficient for the diagonal decomposition and C_A the average. Then an estimation for $\sum$ is calculated using MM for $\sum_0$ and ML for better estimate $\sum$. We use $\sum$ and a fixed β for the MGGD in order to model the window coefficient histogram distribution. Then both the GD and the KLD are going to be used to calculate the information distance between each windows pair and all distances

stored in a similarity matrix. As a final step the spectral clustering method on similarity matrix and the actual classification is done using K-medoids as described in the clustering section.

3. SOFTWARE ENGINEERING DOCUMENTS

3.1. Use Case

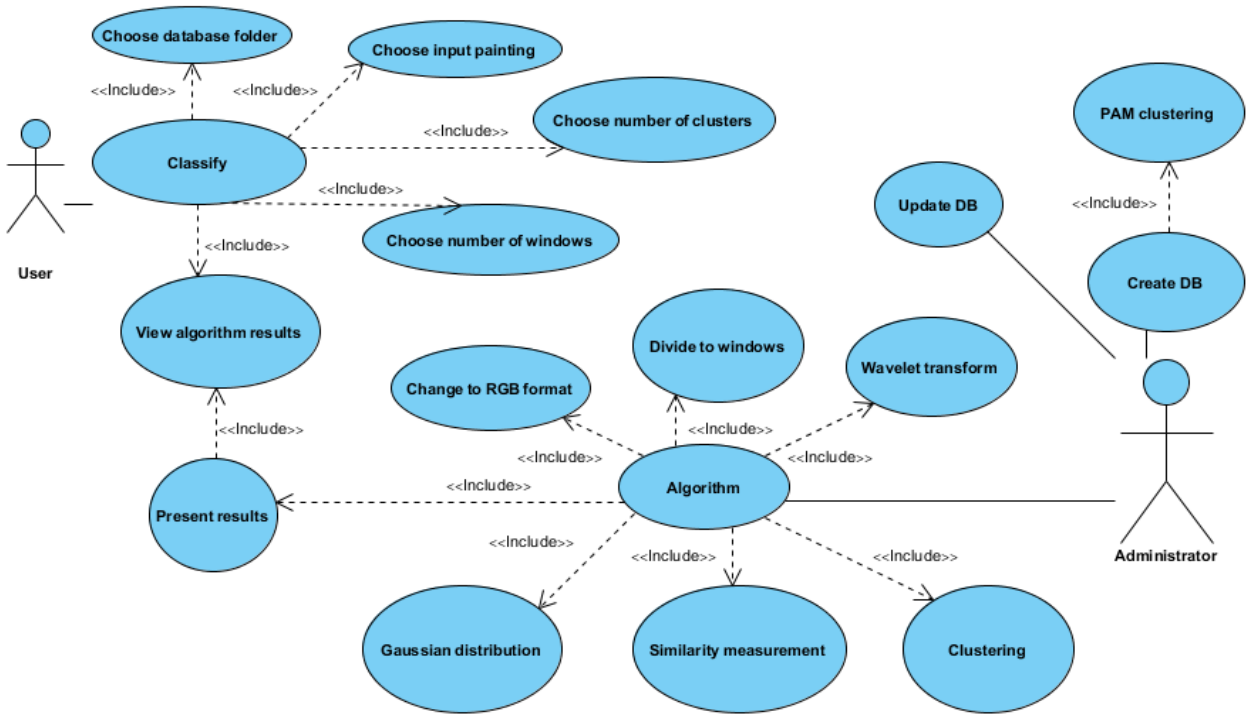


Fig. 9 – Use Case Diagram

3.1.1. Use Case Description

No.	Actor: Administrator	System
	Imports a collection of paintings into a folder.	Saves all paintings as the algorithm database.
	Activates classification option.	Activates the algorithm, database is clustered.
	Update database by add/delete a painting from database.	Adds/deletes a painting and reactivates the algorithm

Table 1. Use Case description of administrator

No.	Actor: User	System
	Activates the application.	Displays main GUI to user with existing database.
	User chooses database folder option.	System takes in account a database of pictures.
	User chooses a painting to classify option.	System gets input from user that will be checked for the algorithm
	User chooses number of clusters and the number of windows the painting will be divided in.	System takes in account the two parameters.
	User chooses to start classification option.	System actives its main algorithm according to users input.
		Algorithm converts a painting to RGB format.
		Algorithm divides the painting to N windows.
		Algorithm uses Wavelet Transform.
		Algorithm uses Gaussian Distribution.
		Algorithm uses Similarity Measurement.
		Algorithm uses Clustering.
	User sees algorithm's results.	Algorithm is done. Presents calculated results to user.

Table 2. Use Case description of user

3.2. Class Diagram

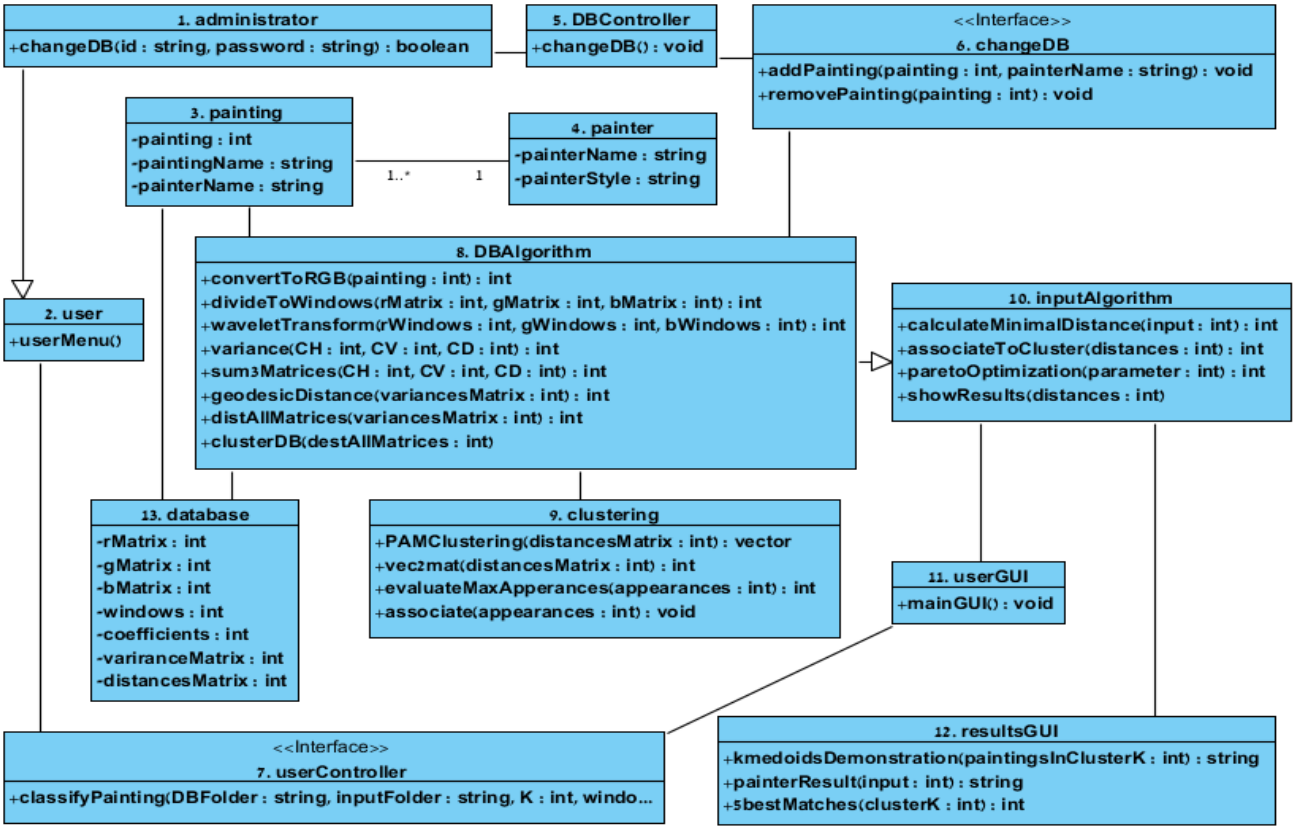


Fig. 10 – Class Diagram

3.3. GUI

3.3.1. GUI: Picture Selection

The user has to select the following:

- Databases folder.
- Input painting as a file.
- Number of clusters.
- Number of windows (must be a square number).
- Choose a database of painters.

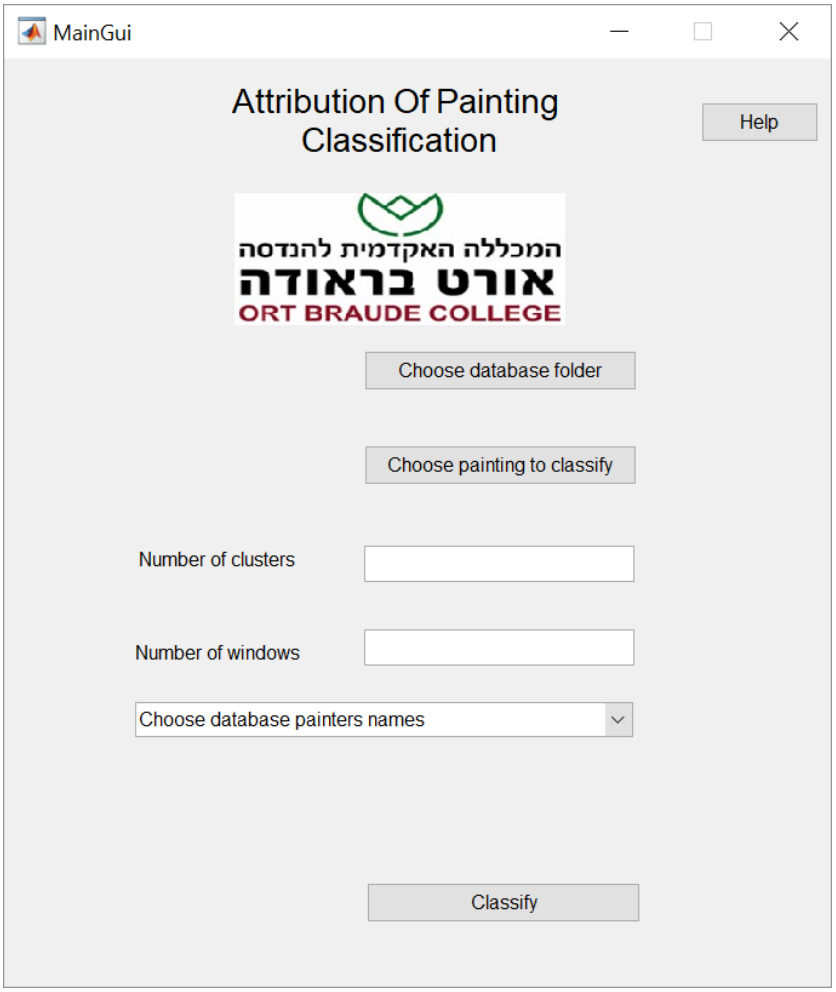


Fig. 11 – Empty main screen

Let us choose data:

- Databases folder.

Fig. 12 – Choosing database folder

Input painting as a file:

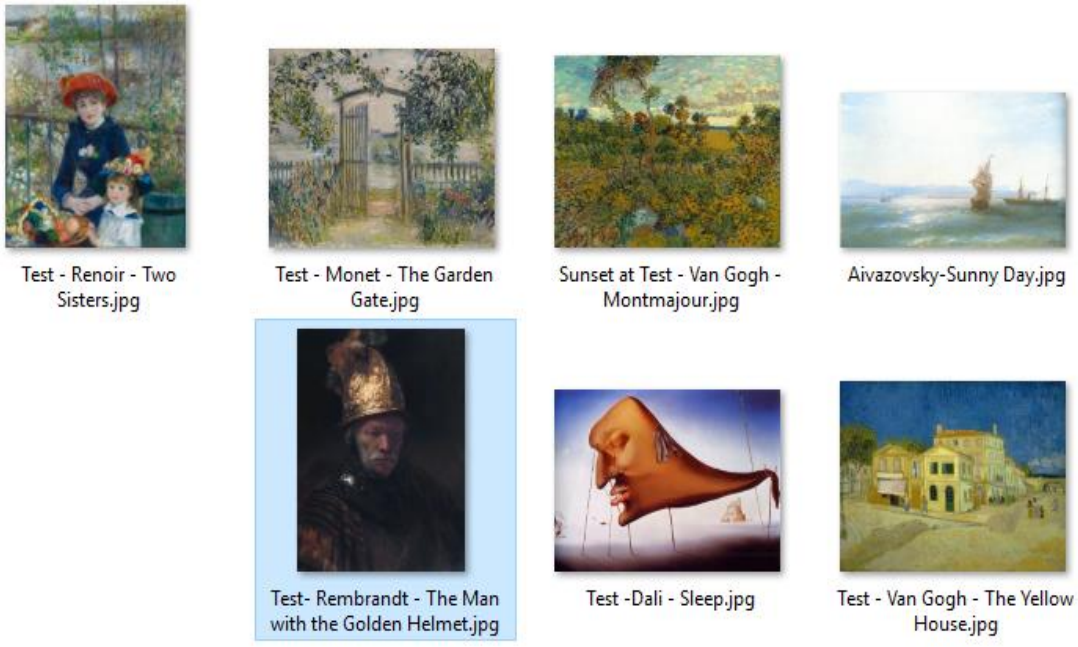


Fig. 13 – Choosing input

- Number of clusters.
- Number of windows (must be a square number).

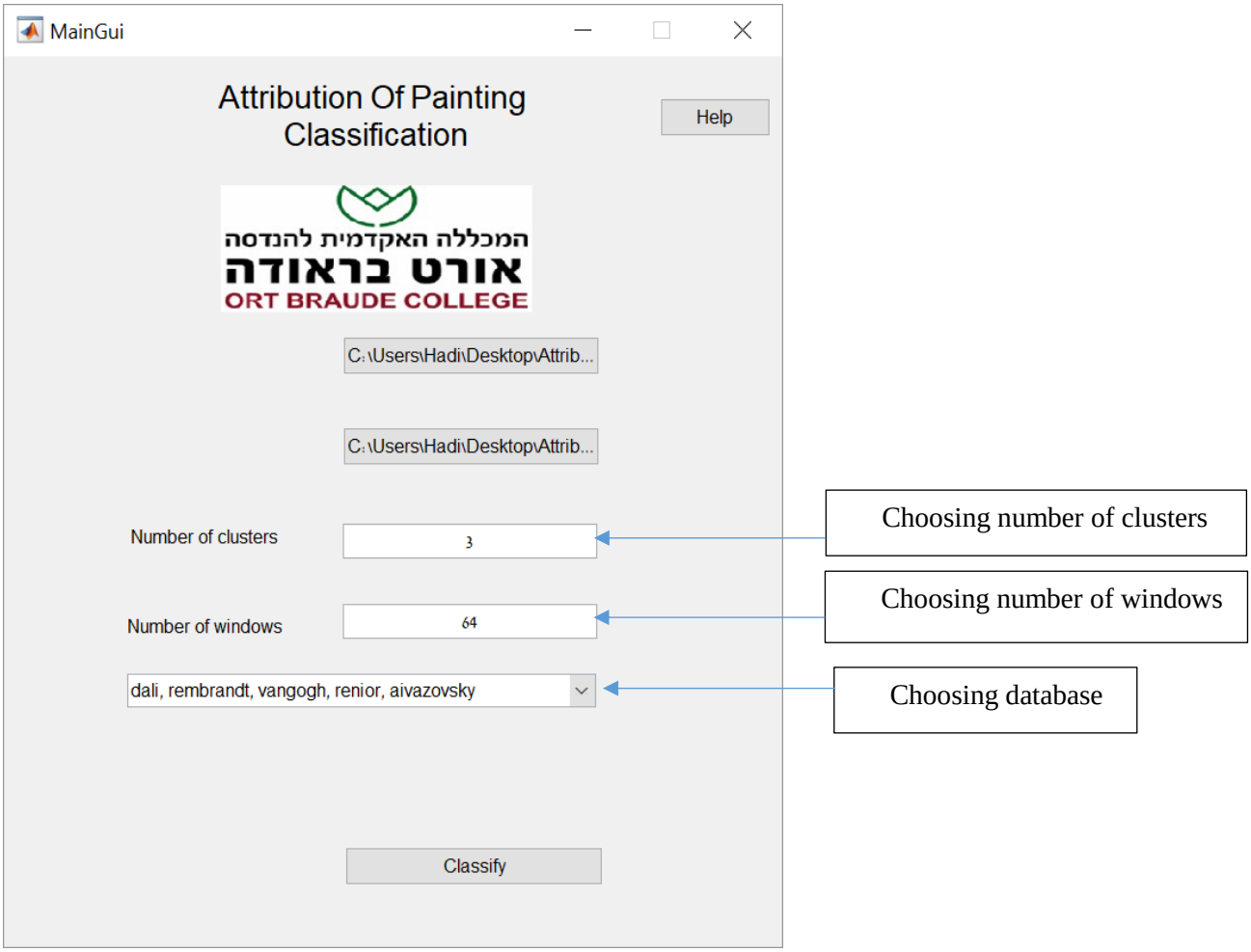


Fig. 14 – Main screen

Now main screen has all necessary data from user and it is ready to run.

3.3.2. GUI Results

After algorithm has completed all calculations, it displays the results to user.

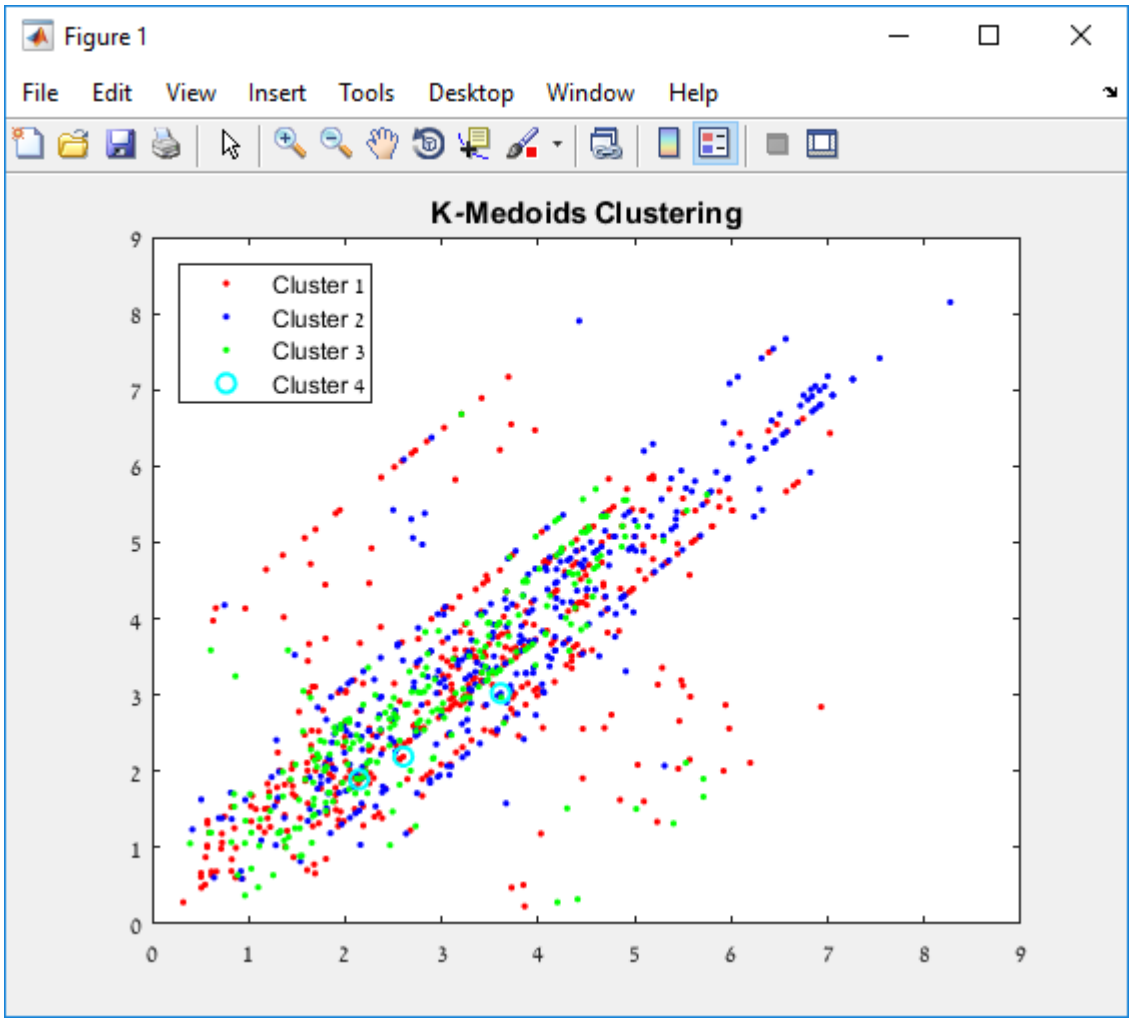


Fig. 15 – K-Medoids Clustering as 2D graph, Cluster 4 are the 3 medoids (k=3)

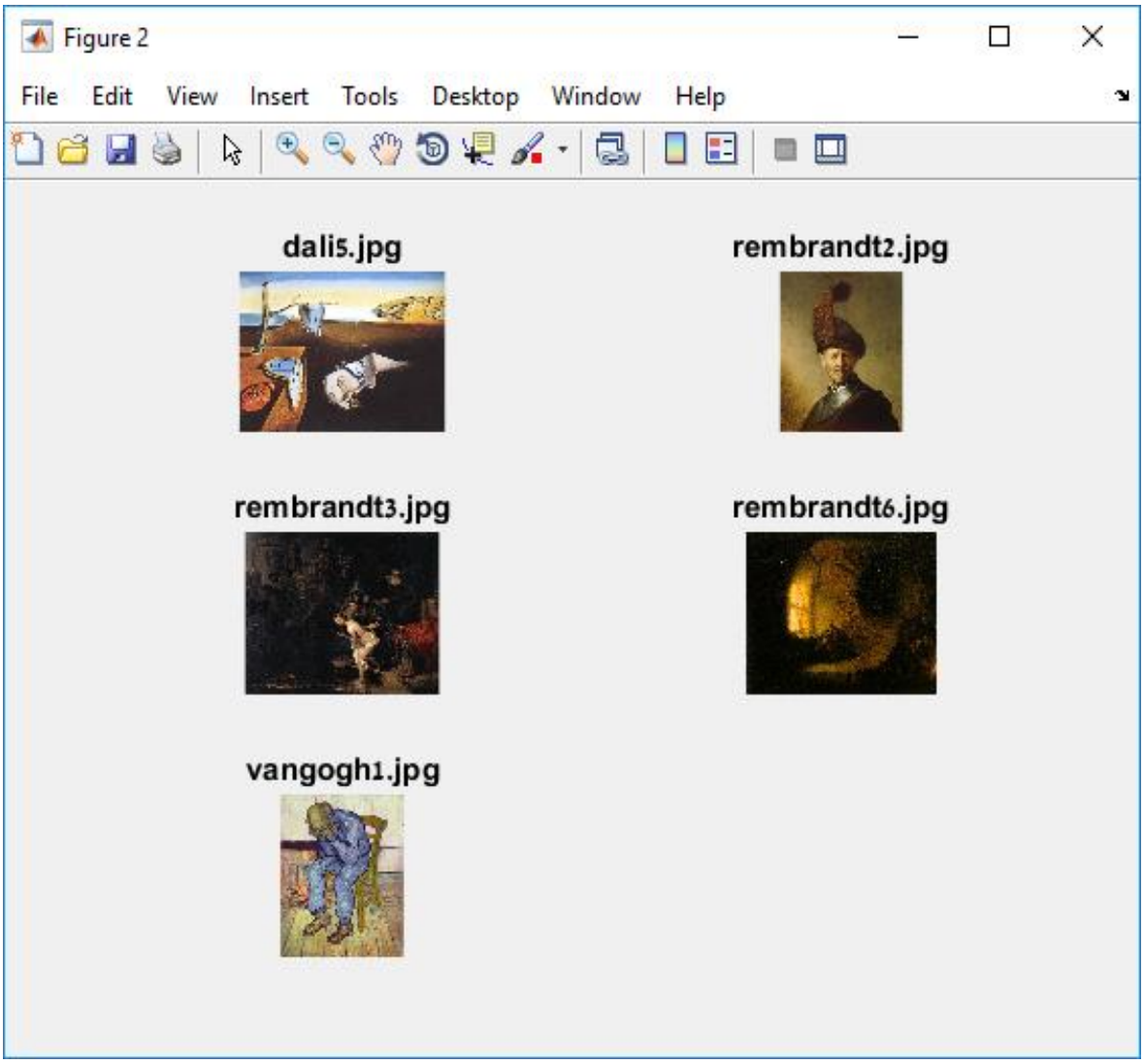


Fig. 16 – Best 5 matches

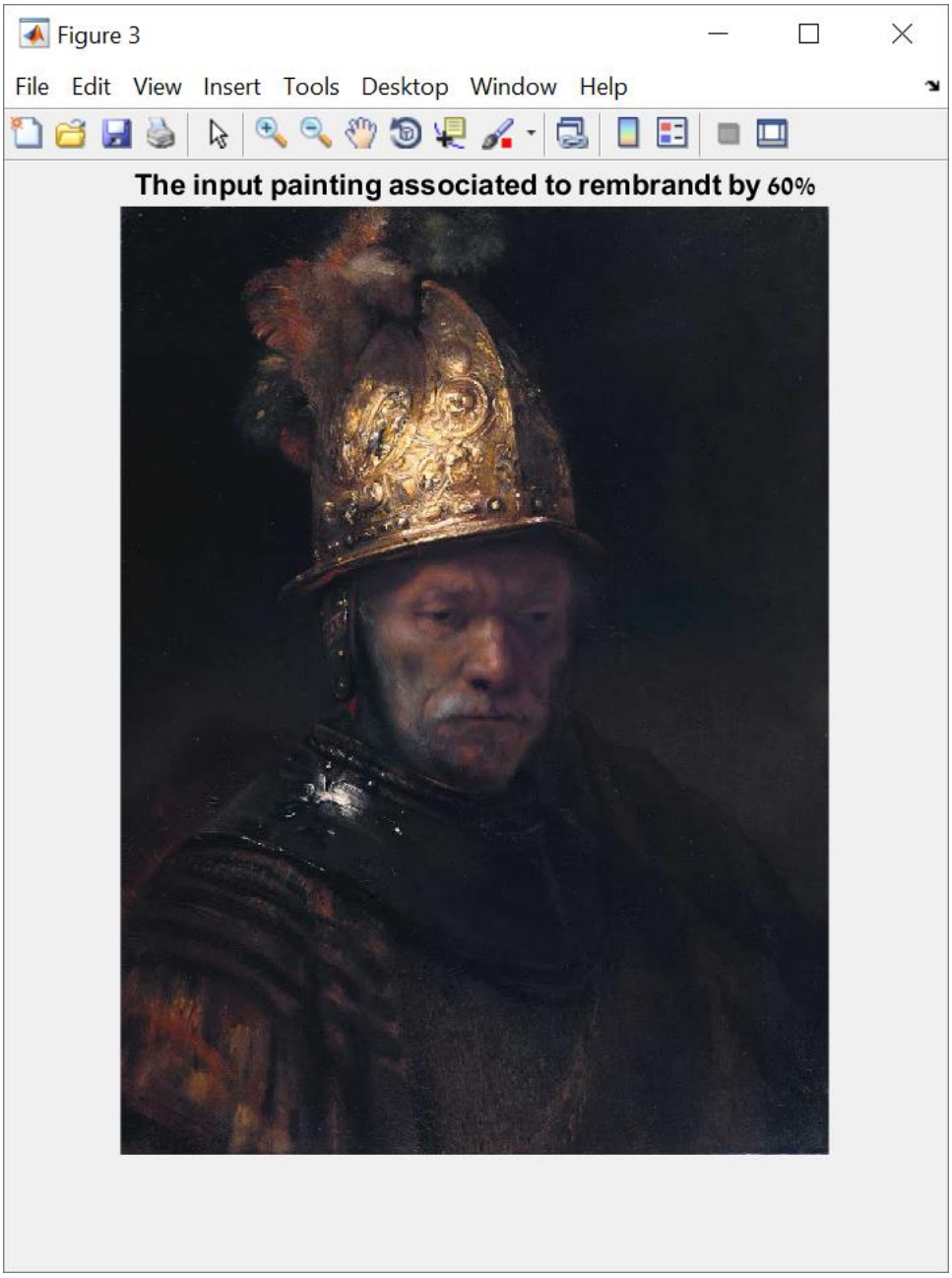


Fig. 17 – Associating a painter to input

The algorithm found that the input is associated to Rembrandt by 60%.

The algorithm found that 3 out of 5 paintings that represents the 5 most suitable paintings in the associated cluster belong to Rembrandt.

These results are very good for the algorithm.

3.4. Testing

In order to check out the system performance we run the program on several significant inputs:

3.4.1. Build Database Testing

Test ID	Description	Expected results	Actual results	Comments
BuildDB 1	Enter: "MainGui" screen Databases folder: C:\Users\Tomer\Downloads\NEW\emptyFolder Inputs folder: C:\Users\Tomer\Downloads\NEW\db1 Number of clusters: 5 Number of windows: 64 Click: "Classify"	System throws out an error	Error	Precondition: The folder c/project/DB must contain paintings. Databases folder is empty.
BuildDB 2	Enter: "MainGui" screen Databases folder: "" Inputs folder: C:\Users\Tomer\Downloads\NEW\testing Number of clusters: 5 Number of windows: 64 Click: "Classify"	System throws out an error	Error	No folder of database is chosen.
BuildDB 3	Enter: "MainGui" screen Databases folder: C:\Users\Tomer\Downloads\NEW\db1 Inputs folder: "" Number of clusters: 5 Number of windows: 64 Click: "Classify"	System throws out an error	Error	No input is chosen.
BuildDB 4	Enter: "MainGui" screen Databases folder: C:\Users\Tomer\Downloads\NEW\db1 Inputs folder: C:\Users\Tomer\Downloads\NEW\testing Number of clusters: 0 Number of windows: 64 Click: "Classify"	System throws out an error	Error	Number of clusters must be a natural number.
BuildDB 5	Enter: "MainGui" screen Databases folder: C:\Users\Tomer\Downloads\NEW\db1	System throws out an error	Error	Number of clusters must be a natural number.

	Inputs folder: C:\Users\Tomer\Downloads\NEW\testing Number of clusters: 0 Number of windows: 64 Click: "Classify"			
BuildDB 6	Enter: "MainGui" screen Databases folder: C:\Users\Tomer\Downloads\NEW\db1 Inputs folder: C:\Users\Tomer\Downloads\NEW\testing Number of clusters: 5 Number of windows: 20 Click: "Classify"	System throws out an error	Error	Number of windows must be a square number.
BuildDB 7	Enter: "MainGui" screen Databases folder: C:\Users\Tomer\Downloads\NEW\db1 Inputs folder: C:\Users\Tomer\Downloads\NEW\testing Number of clusters:5 Number of clusters:64 Click: "Classify"	The algorithm is running	Pass	All preconditions are met.

Table 3. BuildDB Testing Plan

3.4.2. Classification Testing

First database contains 16 paintings:

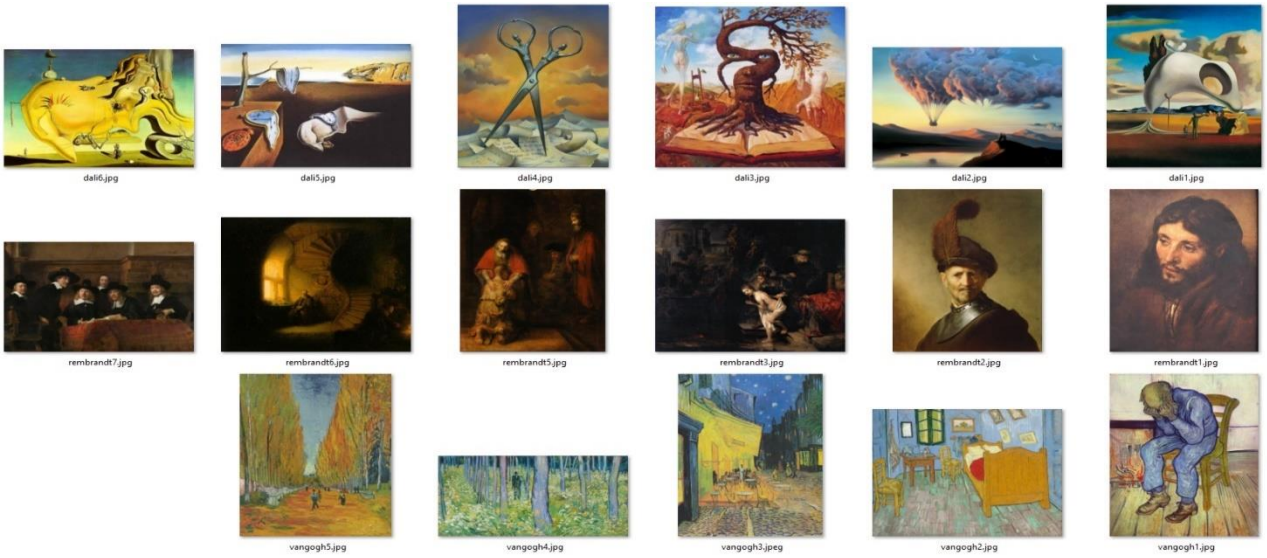


Fig. 18 – "DB1" contains 17 paintings

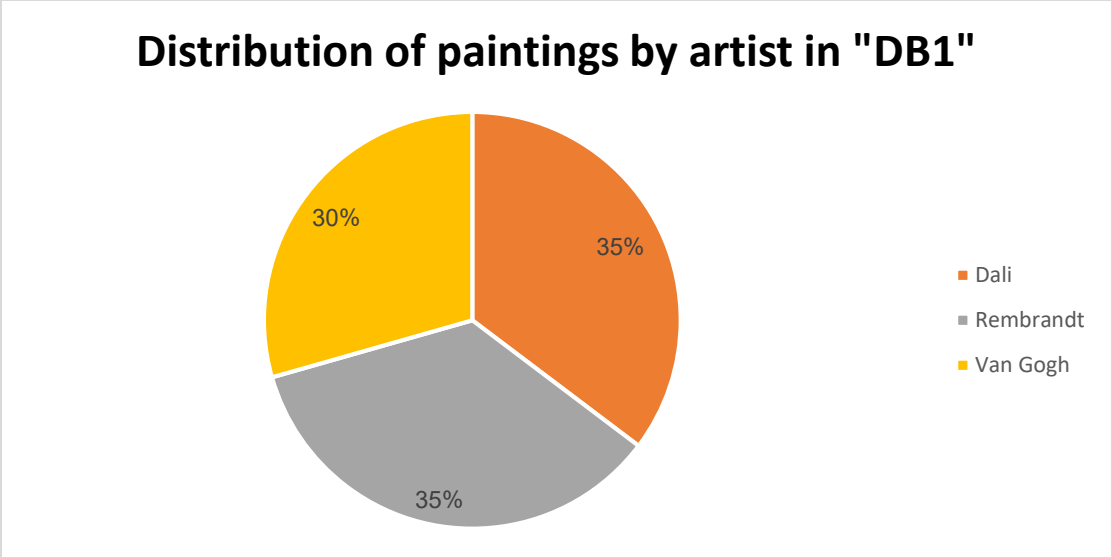


Fig. 19 - Distribution of paintings by painters in "DB1" in pie chart

"DB1" testing out of 10 attempts to van Gogh's painting (Sunset at Montmajour):

Test ID	Description	Actual results	Comments
Classification1	Enter: "MainGui" screen Valid database folder Input: Van Gogh's painting Number of clusters: 3 Number of clusters: 64 Click: "Classify"	100% success	Database is balanced. All runs succeeded.
Classification2	Enter: "MainGui" screen Valid database folder Input: Van Gogh's painting Number of clusters: 3 Number of clusters: 36 Click: "Classify"	70% success	In seven attempts, associated to right painter.
Classification3	Enter: "MainGui" screen Valid database folder Input: Van Gogh's painting Number of clusters: 3 Number of clusters: 16 Click: "Classify"	0% success	All runs failed.
Classification4	Enter: "MainGui" screen Valid database folder Input: Van Gogh's painting Number of clusters: 2 Number of clusters: 64 Click: "Classify"	90% success	In nine attempts, associated to right painter.
Classification5	Enter: "MainGui" screen Valid database folder Input: Van Gogh's painting	80% success	In eight attempts, associated to right painter.

	Number of clusters: 2 Number of clusters: 36 Click: "Classify"		
Classification6	Enter: "MainGui" screen Valid database folder Input: Van Gogh's painting Number of clusters: 2 Number of clusters: 16 Click: "Classify"	10% success	In one attempt, associated to the right painter.

Table 4. DB1 Testing Plan with van Gogh's painting

"DB1" testing out of 10 attempts to Dali's painting (Sleep):

Test ID	Description	Actual results	Comments
Classification1	Enter: "MainGui" screen Valid database folder Input: Dali's painting Number of clusters: 3 Number of clusters: 64 Click: "Classify"	100% success	Database is balaneced. All runs succeeded.
Classification2	Enter: "MainGui" screen Valid database folder Input: Dali's painting Number of clusters: 3 Number of clusters: 36 Click: "Classify"	10% success	In one attempt, associated to the right painter.
Classification3	Enter: "MainGui" screen Valid database folder Input: Dali's painting Number of clusters: 3 Number of clusters: 16 Click: "Classify"	70% success	In seven attempt, associated to the right painter.
Classification4	Enter: "MainGui" screen Valid database folder Input: Dali's painting Number of clusters: 2 Number of clusters: 64 Click: "Classify"	100% success	All runs succeeded.
Classifacation5	Enter: "MainGui" screen Valid database folder Input: Dali's painting Number of clusters: 2 Number of clusters: 36 Click: "Classify"	100% success	All runs succeeded.
Classifacation5	Enter: "MainGui" screen Valid database folder Input: Dali's painting Number of clusters: 2 Number of clusters: 16 Click: "Classify"	100% success	All runs succeeded.

Table 5. DB1 Testing Plan with Dali's painting

Second database contains 45 paintings:

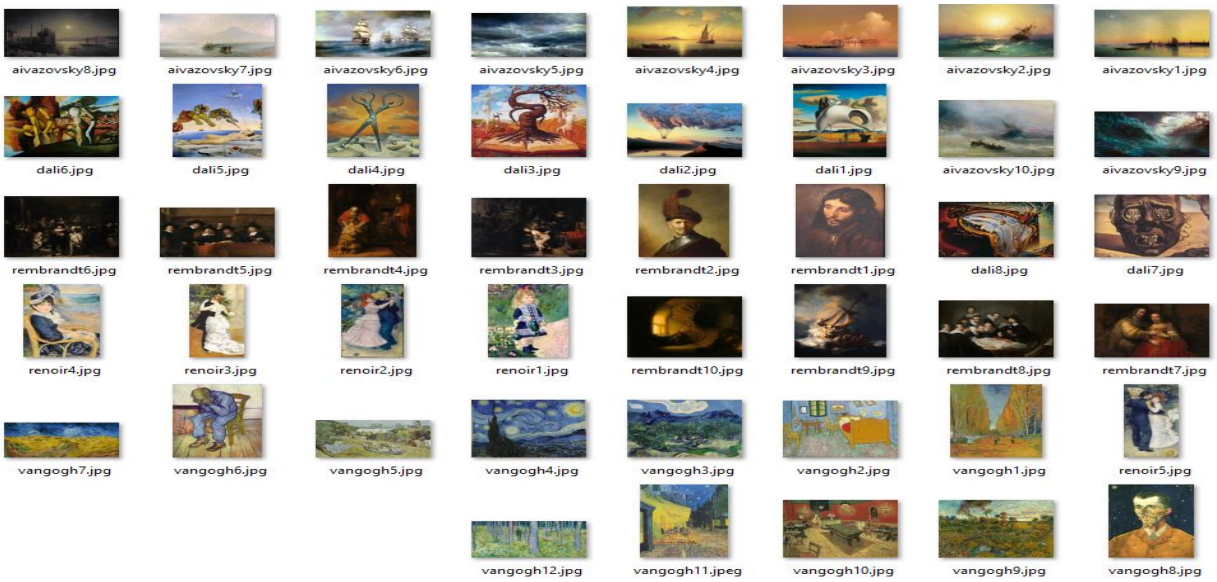


Fig. 20 – "DB2" contains 45 paintings

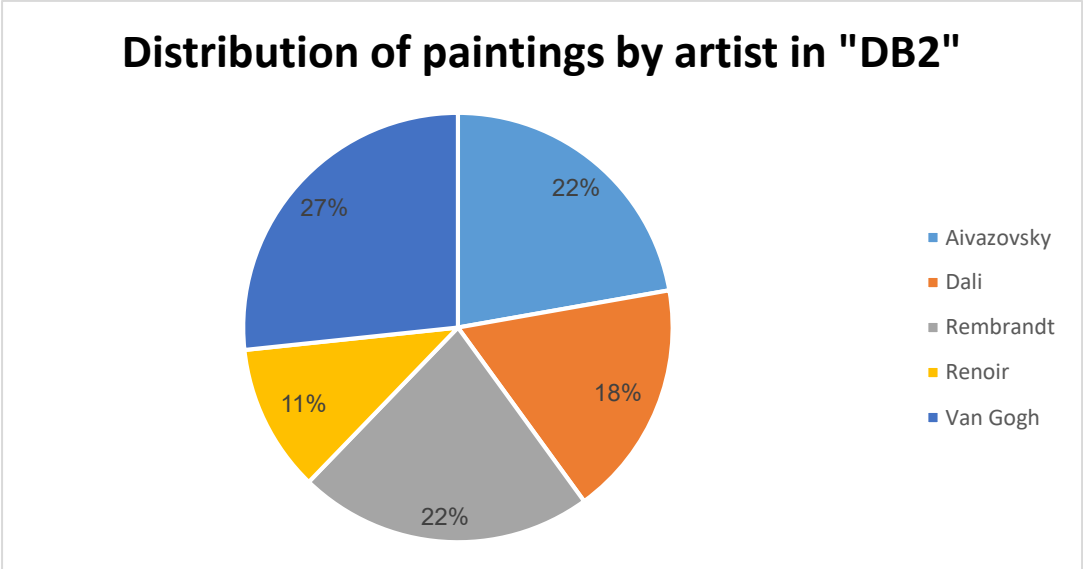


Fig. 21 - Distribution of paintings by painters in "DB2" in pie chart

"DB2" testing out of 10 attempts to van Gogh painting (The Yellow House):

Test ID	Description	**Actual results**	**Comments**
Classification1	Enter: "MainGui" screen Valid database folder Input: Van Gogh's painting Number of clusters: 5 Number of clusters: 64 Click: "Classify"	100% success	Van Gogh has the largest number of paintings in database. All runs succeeded.
Classification2	Enter: "MainGui" screen Valid database folder Input: Van Gogh's painting Number of clusters: 5 Number of clusters: 36 Click: "Classify"	20% success	In two attempt, associated to the right painter.
Classification3	Enter: "MainGui" screen Valid database folder Input: Van Gogh's painting Number of clusters: 5 Number of clusters: 16 Click: "Classify"	50% success	In five attempts, associated to the right painter.
Classification4	Enter: "MainGui" screen Valid database folder Input: Van Gogh's painting Number of clusters: 4 Number of clusters: 64 Click: "Classify"	30% success	In three attempts, associated to the right painter.
Classification5	Enter: "MainGui" screen Valid database folder Input: Van Gogh's painting Number of clusters: 3 Number of clusters: 64 Click: "Classify"	20% success	In two attempts, associated to the right painter.

Classification6	Enter: "MainGui" screen Valid database folder Input: Van Gogh's painting Number of clusters: 3 Number of clusters: 36 Click: "Classify"	0% success	All runs failed.

Table 6. DB2 Testing Plan with van Gogh's painting

"DB2" testing out of 10 attempts to Aivazovsky's painting (Sunny Day):

Test ID	**Description**	**Actual results**	**Comments**
Classification1	Enter: "MainGui" screen Valid database folder Input: Aivazovsky's painting Number of clusters: 5 Number of clusters: 64 Click: "Classify"	80% success	Aivazovsky's amount of paintings is balanced to database. In eight attempts, associated to the right painter.
Classification2	Enter: "MainGui" screen Valid database folder Input: Aivazovsky's painting Number of clusters: 5 Number of clusters: 36 Click: "Classify"	60% success	In six attempts, associated to the right painter.
Classification3	Enter: "MainGui" screen Valid database folder Input: Aivazovsky's painting Number of clusters: 5 Number of clusters: 16 Click: "Classify"	40% success	In four attempts, associated to the right painter.
Classification4	Enter: "MainGui" screen Valid database folder Input: Aivazovsky's painting Number of clusters: 4	50% success	In five attempts, associated to the right painter.

	Number of clusters: 64 Click: "Classify"		
Classification5	Enter: "MainGui" screen Valid database folder Input: Aivazovsky's painting Number of clusters: 4 Number of clusters: 36 Click: "Classify"	20% success	In two attempts, associated to the right painter.
Classification5	Enter: "MainGui" screen Valid database folder Input: Aivazovsky's painting Number of clusters: 3 Number of clusters: 36 Click: "Classify"	0% success	All runs failed.

Table 7. DB2 Testing Plan with Aivazovsky's painting

Third database contains 12 paintings:

Fig. 22 – "DB3" contains 12 paintings

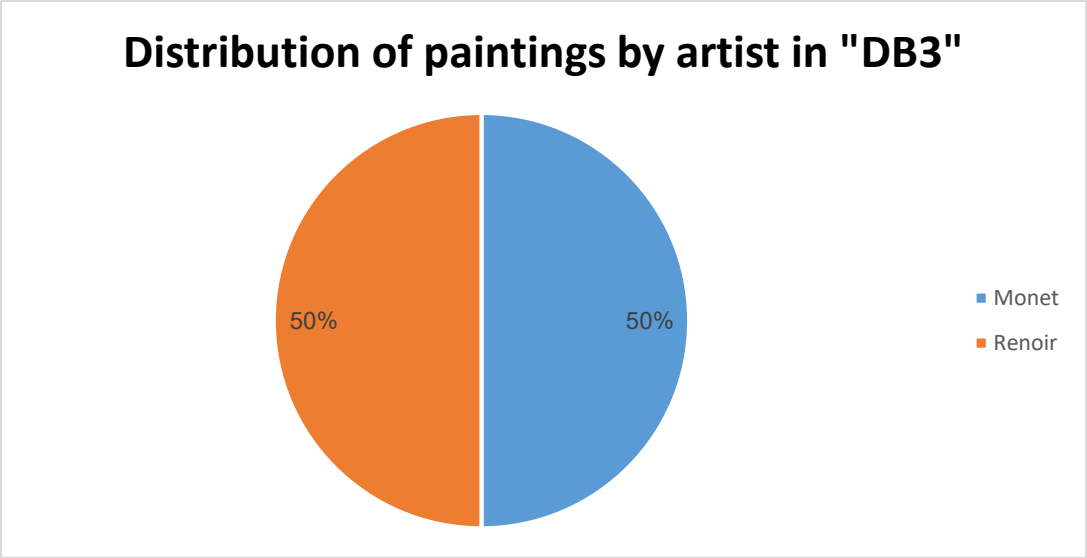


Fig. 23 - Distribution of paintings by painters in "DB3" in pie chart

"DB3" testing out of 10 attempts to Monet's painting (The Garden Gate):

Test ID	Description	Actual results	Comments
Classification1	Enter: "MainGui" screen Valid database folder Input: Monet's painting Number of clusters: 2 Number of clusters: 64 Click: "Classify"	100% success	Database is balanced. All runs succeeded.
Classification2	Enter: "MainGui" screen Valid database folder Input: Monet's painting Number of clusters: 2 Number of clusters: 36 Click: "Classify"	100% success	All runs succeeded.
Classification3	Enter: "MainGui" screen Valid database folder Input: Monet's painting Number of clusters: 2 Number of clusters: 16 Click: "Classify"	100% success	All runs succeeded.
Classification4	Enter: "MainGui" screen Valid database folder Input: Monet's painting Number of clusters: 1 Number of clusters: 64 Click: "Classify"	100% success	All runs succeeded.
Classification5	Enter: "MainGui" screen Valid database folder	100% success	All runs succeeded.

	Input: Monet's painting Number of clusters: 1 Number of clusters: 36 Click: "Classify"		
Classification6	Enter: "MainGui" screen Valid database folder Input: Monet's painting Number of clusters: 1 Number of clusters: 16 Click: "Classify"	100% success	All runs succeeded.

Table 8. DB3 Testing Plan with Monets painting

"DB3" testing out of 10 attempts to Renoir's painting (Two sisters):

Test ID	Description	Actual results	Comments
Classification1	Enter: "MainGui" screen Valid database folder Input: Renoir's painting Number of clusters: 2 Number of clusters: 64 Click: "Classify"	100% success	Database is balanced. All runs succeeded.
Classification2	Enter: "MainGui" screen Valid database folder Input: Renoir's painting Number of clusters: 2 Number of clusters: 36 Click: "Classify"	100% success	All runs succeeded.
Classification3	Enter: "MainGui" screen Valid database folder Input: Renoir's painting Number of clusters: 2 Number of clusters: 16 Click: "Classify"	100% success	All runs succeeded.
Classification4	Enter: "MainGui" screen Valid database folder Input: Renoir's painting	100% success	All runs succeeded.

	Number of clusters: 1 Number of clusters: 64 Click: "Classify"		
Classification5	Enter: "MainGui" screen Valid database folder Input: Renoir's painting Number of clusters: 1 Number of clusters: 36 Click: "Classify"	100% success	All runs succeeded.
Classification6	Enter: "MainGui" screen Valid database folder Input: Renoir's painting Number of clusters: 1 Number of clusters: 16 Click: "Classify"	100% success	All runs succeeded.

Table 9. DB3 Testing Plan with Renoirs painting

4. RESULTS AND CONCLUTIONS

The algorithm works and provides right results in specific cases. The number "K" which represents the number of medoids has to be very accurate according to the number of real number of styles in database. The number of windows for division must not be too small. These two parameters conditions have to be taken in account at the same time. We can see for example that "K"=5 and number of windows=64 provides us excellent results, but if one of them is changed to be smaller (even when the change is minimal), the results are getting wrong in dramatic way.

Naturally, the algorithm uses a big database of many paintings. Each painting divided to windows and the number of comparisons (between each window to each other windows in all other paintings) checked is huge. This huge number has to overcome different paintings techniques and style. It is quite difficult, since many times, even the same artist can have two paintings that do not belong the same style.

Once a combination of K and number of clusters provides us wrong results around 100%, decreasing at least one of them is pointless.

In order to stabilize the algorithm for providing us good results, we choose our database carefully. For most artists, it contains paintings that mostly have similarities. According to our testing, the algorithm odds of success also based on the structure of database.

The first and second databases verify that choosing parameters size is crucial. Even when one of them is selected differentially, the results may turn around in 180 degrees.

Important points that effect the algorithms success rate:

"K" must be accurate to the number of real number of clusters. It makes sense because if the database holds some "N" clusters, clustering database to N-1 clusters will aggregate paintings into wrong cluster and decrease success rate.

The larger number of windows, the better success rate. While the number of windows is large, we check a deeper resolution of paintings.

According to our testing result, we got better results when our input belongs to a painter with number of his paintings in database is near the average.

For example: in both databases we have paintings of 5 painters. In table 5, van Gogh has 1/16 (6%) paintings in database and the algorithm failed in all attempts. In table 6, van Gogh has 12/45 (27%) paintings in database, the algorithm succeed in 9/10 of attempts. Plain to see, the second database is closer to balanced database, that is why the success rate in the second database was significantly higher.

The general conclusion here is: if database has paintings of N painters and the distribution of paintings was divided normally, we expect that each painter has 100/N% of the paintings in database.

Choosing a database not wisely means: the collection of each painter has paintings that do not share common characteristics. This case may associate paintings of the same painter to different clusters, create a random clustering and will decrease the success rate. Moreover, such clustering is not useful.

Choosing a database structure not wisely means: many painters with a low amount of paintings. This case will definitely be risky, even when our algorithm is upgraded.

References

[1] Image Segmentation via Neighborhood Based Similarity. Ehud Kobi, Master of software engineering final research project by Zeev Volkovich supervisor, 2016.

[2] Elena Ravve, Search for similar texts in professional databases project, 2016.

[3] Elena Ravve, Attribution of Paintings previous project, 2015.

[4] Arivazhagan and L Ganesan. Texture segmentation using wavelet transform. Pattern Recognition Letters, 24(16):3197-3203, 2003.

[5] Zeev Volkovich and Renata Avros. Model selection and stability in spectral clustering. In KDIR, pages 25-34, 2012.

[6] Lionel Bombrun, Yannick Berthoumieu, Nour-Eddine Lasmar, and Geert Verdoolaege. Multivariate texture retrieval using the geodesic distance between elliptically distributed random variables. In Image Processing (ICIP), 2011 18th IEEE International Conference on, pages 3637-3640. IEEE, 2011.

[7] Hauz Khas, Aruna Bhat Department of Electrical Engineering, IIT Delhi, Hauz Khas, New Delhi.