Square Packing as Algorithmic Art

Mtech. Thesis submitted in fulfillment of the requirement of the degree of
Master of Technology
In
Computer Science and Engineering
By

Krishnendu Saha
11CS30047

Under the guidance of:

Prof. Partha Bhowmick

Department of Computer Science and Engineering

INDIAN INSTITUTE OF TECHNOLOGY
Kharagpur-721302
INDIA

CERTIFICATE

This is to certify that the thesis entitled *Square Packing as Algorithmic Art* is being submitted by *Krishnendu Saha*, roll no: 11CS30047, to the Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur, India-721302, during the academic session 2015-2016 is a bonafide record of work carried out by him under my supervision and guidance. The report has fulfilled all the requirements as per the regulations of this institute.

Prof. Partha Bhowmick
Department of Computer Science and Engineering,
Indian Institute of Technology,
Kharagpur, India - 721302

Abstract

In this thesis we present a square packing algorithm to tile real images as a form of digital art.Main objective of the tiling process is to retain features of the actual image as much as possible using less number of squares.The packing algorithm resembles the greedy approach of solving coin denomination problem.To aid the packing problem three different distance metric (Manhattan, Chebyshev and Euclidean) are compared for best results.Different distance mapping techniques are looked into to make the packing algorithm efficient. To remove unnecessary detailing of image, retaining the visually significant features we resort to meanshift clustering. Alignment of squares is also taken account to increase packing density and artistic appeal. Overall this algorithm gives a tiled, less detailed, but still recognizable form of the original image.

Contents

List of Figures

1. Introduction:

Algorithmic art, mostly visual art is a subset of generative art, in which there works an autonomous non-human process to determine the features of art. Specifically a design to be considered as algorithmic art it needs to be created only by an algorithm without any or negligible human supervision. Algorithms being deterministic it will create identical artworks under same conditions and inputs .Variability may be introduced by pseudo random numbers .These algorithms often try to imitate different techniques used by human artists and often succeed to do so in much less time than a human with higher accuracy. Even the simplest algorithms takes too much manual calculation for an image of reasonable size. Hence computers are being used to explore new techniques of art that needs intensive calculations.

Historically, much before the advent of computers, algorithms and mathematical techniques have been used to create art. Roman Verostko argues that Islamic geometric patterns are constructed using algorithms, as are Italian Renaissance paintings which make use of mathematical techniques, in particular linear perspective and proportion [1]. Some of the earliest known examples of computer-generated algorithmic art were created by Georg Nees, Frieder Nake, A. Michael Noll, Manfred Mohr and Vera Molnár in the early 1960s. These artworks were executed by a plotter controlled by a computer. As computers became more accessible to artists in the 1970's and 1980's some artists began to experiment with algorithmic procedure. Such digital artists, also called Algorists formally began correspondence and establishing their identity as artists following a panel titled "Art and Algorithms", co-founded by Roman Verostko and Jean-Pierre Hébert at SIGGRAPH in 1995 [2].

1.1. Related Works:

Tessellation is a similar field of work as square packing presented here, which is tiling a 2 dimensional surface with tiles of different shapes without any gaps or overlaps.Only three regular polygons (triangle, square and hexagon) can tesslate a euclidean plane as multiple polygons meeting at point should have an internal angle sum of $360°$(e.g. for square $90° \times 4 = 360°$). One example of such work is of mathematician Roger Penrose's Penrose Rhombus tiling [3].As opposed to tessalation programs which is complete coverage of given surface packing problems deals maximum possible coverage of a container space with given shapes.Some such packing problems in euclidean space are packing circles in squares ,circles in triangles etc. .Many mathematicians calculated the maximum possible packing in many such regular cases. Such an example is, work done by mathematician Erich Friedman as minimum dimension of square, equilateral triangle, required to contain n unit circles or triangles [4].

Square packing problem has been approached to solve many practical problems (e.g. stock cutting, VLSI design, advertisement placement, image processing, and scheduling) [5] [6]. Some of these problems are fundamental geometric problems studied by discrete geometry community [7]. For example one square packing problem is to pack a subset of squares, each having some given profit, from given set so that the profit is maximum [8]. Another problem in discrete geometry regarding square packing is to find the smallest ε such that the reciprocal squares $(\frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \dots)$ can be packed in a rectangle of area $(\pi^2/6 - 1) + \varepsilon$ since it is noted that $\sum_{i=2}^{\infty} 1/i^2 = (\pi^2/6 - 1)$ [9]. In algorithmic art field circle packing algorithm proposed in Digital Circlism as Algorithmic Art [10] is a similar kind of work.

1.2. Organization:

This thesis is organized as follows:

- Chapter 2: Describes the basic square packing problem.
- Chapter 3: Describes the mean shift clustering technique used here to segment the image.
- Chapter 4: Describes different distance metrics and different distance mapping techniques used to aid the square tiling process.
- Chapter 5: Describes the square packing algorithm, some heuristics used to make the tiling efficient,the distacne recalculation and time complexity of the algorithm.
- Chapter 6: Shows the results using different parameters.
- Chapter 7: Concludes the thesis giving some pointers to improve the algorithm.

2. Square Packing:

Packing problems are a branch of optimization problems in which the desired goal is to reach a higher packing density ratio ρ , i.e. fraction of area covered by elementary shapes inside a particular bounded region, without overlapping of elementary shapes used to cover the region. Square packing, loosely called as tiling, is a kind of such packing problems. The problem of square packing is researched by many mathematicians but accurate solution is known for only some simple cases. In fact it is known that most of the packing problems are NP-hard.

In this thesis we consider tiling of any rgb coloured image with different sized monochromatic squares placed in different orientation without overlapping one another. The objectives considered are high packing density (say objecive1) as well as similarity of the tiled image with the original image (objective2). Intuitively we may consider minimum number of squares used will eventually lead to a high packing density as space of a single line needs to be left to make two boundary sharing square distinguishable.So there is a tradeoff required between the two objectives. Though this balance between the two objectives may be parameterised in the algorithm, but the artistic evaluation is ultimately left to human judgement. The steps taken to solve the problem may be divided in three parts, (1) clustering of pixels based on colour and spatial information, (2) distance calculation of each pixel from the boundary of its own cluster, (3) placing of squares based on the distance information of each pixels. In clustering step number of clusters may be varied.Less number of clusters may fail to capture the details of an image on the other hand many small clusters will require many small squares leading to reduced packing as mentioned before. So out of the three major steps clustering takes maximum human judgement in this

algorithm.Meanshift clustering algorithm is used in this case as it may be parameterised to control the degree of clustering (ie. number of clusters). In distance calculation step three different distance parameters are considered: (1) cityblock (or Manhattan) distance, (2) chessboard (or Chebyshev) distance, (3) Euclidean distance. We show here different distance metrics being more suitable in case of different square orientations.

3. Clustering and Segmentation:

Mean Shift is used as a discontinuity preserving filter to cluster the image into meaningful segments based on colour information. In this section a brief review of the algorithm is given based on MS procedure based on [11].It is a nonparametric clustering technique which does not require prior knowledge of the number of clusters, and does not constrain the shape of the clusters.

Given n data points $x_i , i = 1, \dots , n$ on a d-dimensional space R_d, the multivariate kernel density estimate obtained with kernel K(x) and window radius h is

$$f(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

We consider radially symmetric kernels satisfying $K(x) = c_{k,d} k(\|x\|^2)$, where $c_{k,d} > 0$ is a normalization constant such that

$$\int_0^\infty K(x)dx = \int_0^\infty c_{k,d} k(\|x\|^2)dx = 1,$$

$k(x)$ needs to be a monotonically decreasing function necessary for convergence of this procedure. The modes of the density function are located at the zeros of the gradient function $\nabla f(x) = 0$.

The gradient of the density estimator is

$$\nabla f(x) = \frac{2c_{k,d}}{nh^{d+2}} \sum_{i=1}^{n} (x_i - x) g\left(\left\|\frac{x - x_i}{h}\right\|^2\right)$$

$$= \frac{2c_{k,d}}{nh^{d+2}} \left[\sum_{i=1}^{n} g\left(\left\|\frac{x - x_i}{h}\right\|^2\right)\right] \left[\frac{\sum_{i=1}^{n} x_i g\left(\left\|\frac{x - x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{x - x_i}{h}\right\|^2\right)} - x\right]$$

where $g(x) = -k'(x)$. The first term is proportional to the density estimate at x computed with kernel $G(x) = c_{g,d} g(\|x\|^2)$ and the second term

$$m_{h,g}(x) = \frac{\sum_{i=1}^{n} x_i g\left(\left\|\frac{x-x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{x-x_i}{h}\right\|^2\right)} - x$$

is the mean shift. The mean shift vector always points toward the direction of the maximum increase in the density.

The mean shift procedure, obtained by successive

- computation of the mean shift vector $m_h(x_t)$,
- translation of the window $x_{t+1} = x_t + m_h(x_t)$

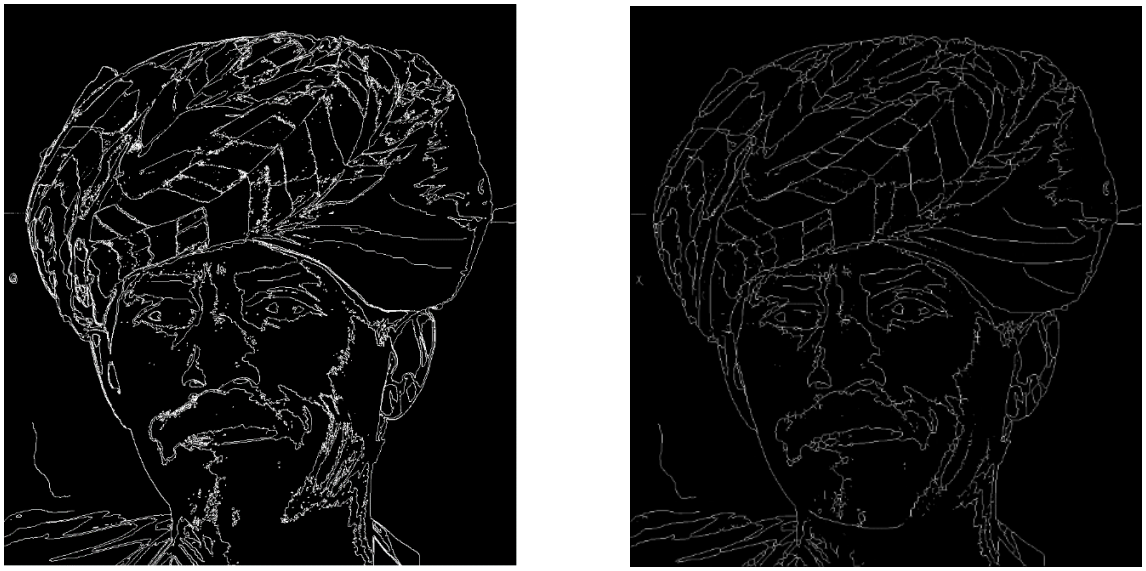
It is guaranteed to converge to a point where the gradient of density function is zero .

This process is applied on a d=p+2 dimensional space, p=3 for colour images, on a joint spatial-range domain.This segmentation process is actually a merging process of the image pixels. This algorithm requires parameter h, which determines the resolution of mode detection. The stopping condition of the meanshift algorithm is given as average shift of the pixels means drop below a threshold parameter value. Decrease in the parameter value may cause in increase of number of clusters as shown in Figure 1. Each pixel is assigned to a cluster with minimum distance from cluster mean.

(a) (b) (c)

Figure 1:Mean Shift (a) Original Image, Mean shift results with different threshold parameter value (b) 0.2, (c) 0.1

After clustering, segmentation of image is done by following boundary between clusters .This segmentation lines contains a lot of noise. To eliminate the noisy lines some morphological operations (like morphological closing, skeletonization etc.) are done done on the binary form of segmentation outline image as shown in Figure 2.



(a) (b)

Figure 2: Segmentation outline image (a) before morphological processing, (b) after morphological processing

4. Distance Transform:

Distance transform, also known as distance mapping is a derived representation of an image.The map labels each pixel of a image with the nearest distance to the nearest obstacle pixel. In our case this obstacle pixel is the boundary of a binary image.

Here distance mapping is applied on the segmentation outline diagram. Segmentation outline is a binary image with two set of pixels

S = set of 1 's ,the inner cluster points and

S' = set of 0 's , the segmentation outline

a distance map L(S) is an image such that for each pixel

(i,j)ϵS , L(i, j)=min(distance((i,j),S')).

This distance gives an approximate measure of maximum size of square that may be fitted from each pixel. Distance mapping depends on the metrics used.

4.1. Distance Metrics:

Here we try three different distance calculation metric, namely

(a)*chessboard or 8 connected distance* , $d_8((i,j),(h,k))=max(|i-h|,|j-k|)$,

(b)*cityblock or 4 connected distance,* $d_4((i,j),(h,k))=|i-h|+|j-k|$,

(c)*euclidean distance,* calculated as $d_e((i,j),(h,k))=\sqrt[2]{(i-h)^2+(j-k)^2}$,

where (i,j) and (h,k) is the x and y coordinate of two different pixels.

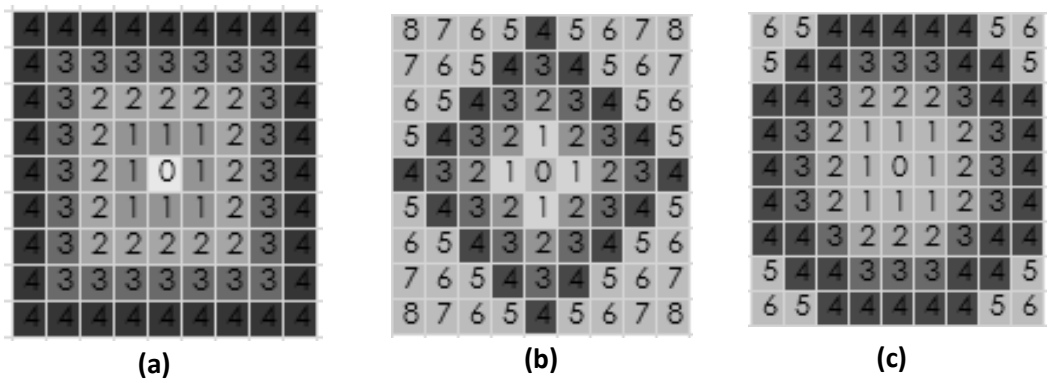


Figure 3 : Distance Contour of (a) chessboard, (b) cityblock, (c) euclidean distance metric

Three different metrics gives 3 different kinds of contour which makes square fitting of a partcular alignment easier for each of the three cases. Chessboard metric gives axes aligned square contour, cityblock metric gives tilted square contour and euclidean metric gives circular contour as shown in Figure 3. This observation helps us hypothesize that d_8 metric distance calculation fits axes aligned squares better and likewise d_4 metric fits 45° rotated squares better and euclidean metric fits square of any alignment.The distance mapping on segmented image is shown in Fig. 4.

4.2. Parallel vs Sequential Distance Mapping:

Distance transform algorithms may be processed in two ways:

(a) Parallel and (b) Sequential.

In parallel algorithms the mapping is done in a single pass, starting from set of segmentation outline points S_0 (i.e 0 distance points) neighbour finding function $L'(S_0)= S_1$, gives the distance 1 points and it is iterated until all

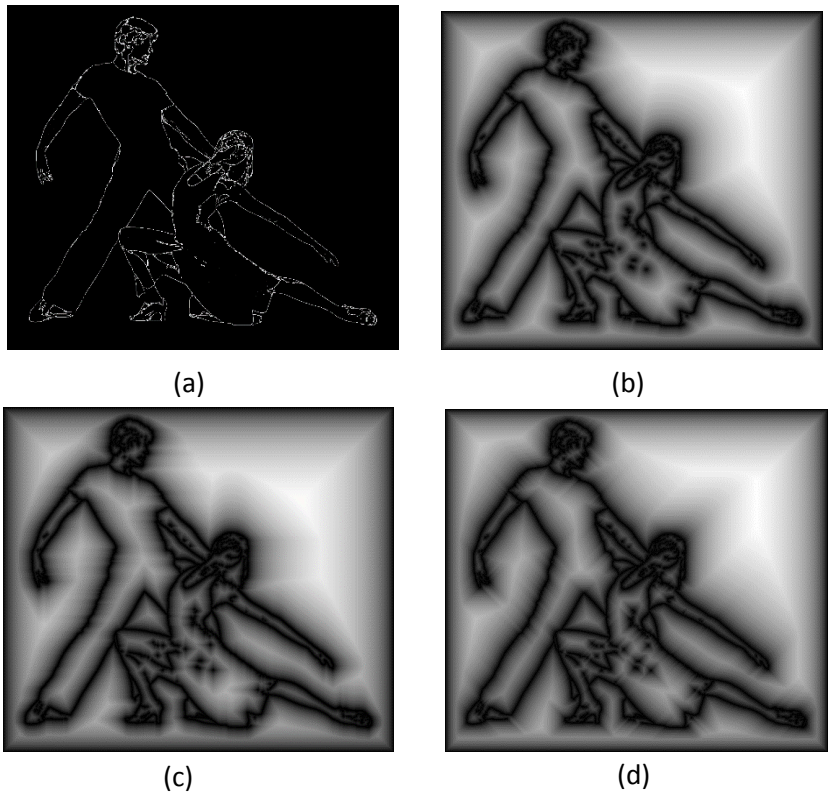


Figure 4: Distance Transforms, (a) Segmented Image, (b) d_8 distance transform, (c)d_4 distance transform, (d) Euclidean distance transform

Algorithm 1 : 8 Sequential Euclidean Distance Transform

The picture L is a two-dimensional array with the elements

$$L(i,j) \qquad 0 \le i \le M-1, 0 \le j \le N-1.$$

Each element is a *two-element vector*

$\bar{L}(i,j) = (L_i, L_j)$, Li, Lj being positive integers,

$L(i,j) = (L_i(i,j), L_j(i,j))$.

The size of a vector *L(i,j)* is defined by

$$|L(i,j)| = \sqrt{Li^2 + Lj^2}$$

8SED:

Initially:

$L(i,j) = (0,0)$ for $(i,j) \in S$

$L(i,j) = (Z, Z)$ for $(i,j) \in S'$

S here denotes the background and S' the segmentation outline

First picture scan:

for j= 1,2 . . . . . N - 1,

for i = 0,1,2 . . . . . M- 1

$\bar{L}(i,j) = \min(\bar{L}(i,j), \bar{L}(i-1,j-1)+(1,1), \bar{L}(i,j-1)+(0,1), \bar{L}(i+1,j-1)+(1,1))$

for i = 1,2 . . . . . M- 1

$\bar{L}(i,j) = \min(\bar{L}(i,j), \bar{L}(i-1,j)+(1,0))$

for i = M-2,M-3 1,0

$\bar{L}(i,j) = \min(\bar{L}(i,j), \bar{L}(i+1,j)+(1,0))$

Second picture scan:

for j= N-2,N-3, ... 1,0

for i = 0,1,2 . . . . . M- 1

$\bar{L}(i,j) = \min(\bar{L}(i,j), \bar{L}(i-1,j+1)+(1,1), \bar{L}(i,j+1)+(0,1), \bar{L}(i+1,j+1)+(1,1))$

for i = 1,2 . . . . . M- 1

$\bar{L}(i,j) = \min(\bar{L}(i,j), \bar{L}(i-1,j)+(1,0))$

for i =M-2 ,. . . . 1,0

$\bar{L}(i,j) = \min(\bar{L}(i,j), \bar{L}(i+1,j)+(1,0))$

The euclidean distance of any point may be calculated as $\sqrt{L_i^2 + L_j^2}$

---

the points are visited . So the mapping function may be given as

$$S_0 \rightarrow L'(S_0) = S_1 \rightarrow L'(S_1) = S_2 \rightarrow \ldots \rightarrow L'(S_{n-1}) = S_n = L(S).$$

In sequential algorithms the mapping is done in multiple passes. In the n^{th} pass $L_n(i,j)$ is calculated from the previous mapping $L_{n-1}(i-1,j)$, $L_{n-1}(i,j-1)$ etc. information.The 8 Sequential Euclidean Distance (8 SED) transform algorithm [12] is given in *Algorithm 1*. The 8 SED algorithm gives a bound of each error as 0.076. In our case maximum error possible is 1 as the distance mapping is in integer space. The min() function in *Algorithm 1* gives the tuple with minimum of sum of square of tuple elements.

Though sequential algorithm take multiple pass compared to single pass of parallel algorithm, sequential algorithm has less memory requirement .Here we use parallel algorithms to calculate chessboard and cityblock distance and sequential algorithm (8SED) [12] in case of euclidean distance. This distance mapping values does not always give a proper estimate of square fitting and in those cases overlapping with other square or boundary overlapping may occur. Hence those overlapping squares needs to be discarded. Both the parallel and sequential methods are O(n), n being total number of pixels in image.

5. Square Fitting:

The distance map gives an estimate of biggest square fitting possible at any pixel. As mentioned earlier different distance maps gives best estimation for square fitting of different alignment.Also the size of square varies in different mappings. Here we consider the size of square as the size of half diagonal denoted by ***r***. The relation between distance mapping value at pixel ***(i,j)*** denoted by ***d*** and largest fitting square size r_l is different in different mappings. In case of d_4 and d_e, r_l is taken as d_4 and d_e respectively ,i.e. $r_l(i,j) = d_4(i,j)$ or $r_l = d_e(i,j)$) but in case of

Algorithm 2 : Square Packing

```
Input: Image I, Image M, Set D
Output: Set S
begin
    for each d in D
        α ← assign_angle();
        r ← k * d
        {sq_boundary, sq_internal} ← precalculate_square(r,α)
        while M(d) is not null
            P ← M(d).get_first();
            bool f=false;
            for p in sq_boundary
                if (M(P+p) is 0 )
                    f = true;
                end_if
            end_for
            if(f is true or variation_abv_threshold(I,sq_internal,P,threshold))
                continue;
            end_if
            bool f1=draw_square(P,sq_internal);
            if(f1)
                S ← S ∪ Sq(P,I,r,α);
                M({sq_internal})←0;
            distance_recalculation(P,sq_boundary,M);
            end_if
        end_while
    end_for
end
```

d_8 it is multiplied by $\sqrt{2}$ i.e. $r_l(i,j) = \sqrt{2} \times d_8(i,j)$. The square fitting is done as *greedy coin denomination* problem solving i.e. square of largest denomination that fits a segment is fitted and this process is iterated on remaining untiled space untill no space is left to fit a square of even the smallest denomination.The denomination set may be chosen randomly with a particular ratio between them or set of all positive integers. The choice is varied to get the maximum visual appeal. Some heuristics are used while square fitting to get maximum packing or visual appeal. Two such heuristics are mentioned in 5.1 .

Algorithm 2 describes the square packing algorithm. It takes as input the actual image $\boldsymbol{I}$, the distance mapping of segmented image as $\boldsymbol{M}$ (obtained from any of the three mentioned distance mapping) and set of square denominations $\boldsymbol{D}$ as user input. Output is provided as a set of squares $\boldsymbol{S}$ with their colour information. The denominations in set $\boldsymbol{D}$ are considered in decreasing order. For each of those denominations, $d \in \boldsymbol{D}$, points are searched from the distance map $\boldsymbol{M}$, denoted as $\boldsymbol{M(d)}$. Suppose $\boldsymbol{P}$ is a pixel in $\boldsymbol{M}$ and $\boldsymbol{P} \in \boldsymbol{M(d)}$, then $\boldsymbol{P}$ gives an suitable point for square fitting of half diagonal length $\boldsymbol{r}=\boldsymbol{k}\times\boldsymbol{d}$. As mentioned earlier in case of d_e, d_4 distance mapping this k is 1 and in case of d_8, $\boldsymbol{k}$ is $\sqrt{2}$. The angle between any diagonal and y axis, denoted as $\boldsymbol{\alpha}$ is varied to give different alignment to the squares. The *assign_angle()* function also based on the distance mapping assigns different values to $\boldsymbol{\alpha}$, 0° in d_4 , 45° in d_8 and it is chosen out of a set of angles $(0°, 15°, 30°, 45°, 60°, 75°)$ in case of d_e .This choice of angle $\boldsymbol{\alpha}$ may done randomly or using some heuristics (e.g. Best Fit Angle calculation) as mentioned in 5.1 . Next the *precalculate_square(r,α)* function computes all square internal, denoted as *sq_internal* and boundary points, denoted as *sq_boundary* relative to centre point of a square. Later if we need to put a square of parameters $\boldsymbol{r}, \boldsymbol{\alpha}$ at pixel $\boldsymbol{P}$ the actual points of squares may be found by adding $\boldsymbol{P}$ and $\boldsymbol{p}$, where p ∈ sq_internal. The *variation_abv_threshold(I,sq_internal,P,threshold)* function calculates the colour variance of the actual image I in square area that is going to

be fitted. If variation is not above a predefined threshold then the square is actually drawn, After the square is drawn the distance mapping is done again by *distance_recalculation()* function. Finally the algorithm returns the set of squares ***S*** where each square vertices and colour are calculated by function *Sq(P,I,r,α).*

5.1. Heuristics Used in square fitting:

- *Checking the colour variance*: Though the image is clustered based on colour similarity of pixels, for further refinement a colour variance checking is done before fitting a square. While fitting a square the colour variance of the actual image within the square must be low. Squares with high colour variance are then discarded. This helps us fit more number of smaller squares in detailed regions.

- *Best Fit angle calculation:* In case of euclidean distance metric d_8 the orientation of the square may be varied to give the best fitting locally. The angle of the diagonal with the horizontal is varied from 0^0 to 75^0 with step 15^0. The best possible fit out of those six cases is decided by a function. Let for an angle α, $S \epsilon Foreground\ points$ be the set of points for which the euclidean distance is going to be changed.Then the sum of the differences is calculated normalized by the number of total points as

$$\sigma_\alpha = (\sum_{s \epsilon S} |\partial_s - \delta_s|)/|S|$$

Here ∂ and δ are the euclidean distances before and after the square is put. Now a α is selected for which σ_α is minimum.The hypothesis behind this is to keep intact more points with greater mapping distance points so that bigger squares may be fitted later.

5.2. Local Distance recalculation:

After each square fitting, the distance map changes. The square tiled regions may be considered as holes in image and the square boundary as new

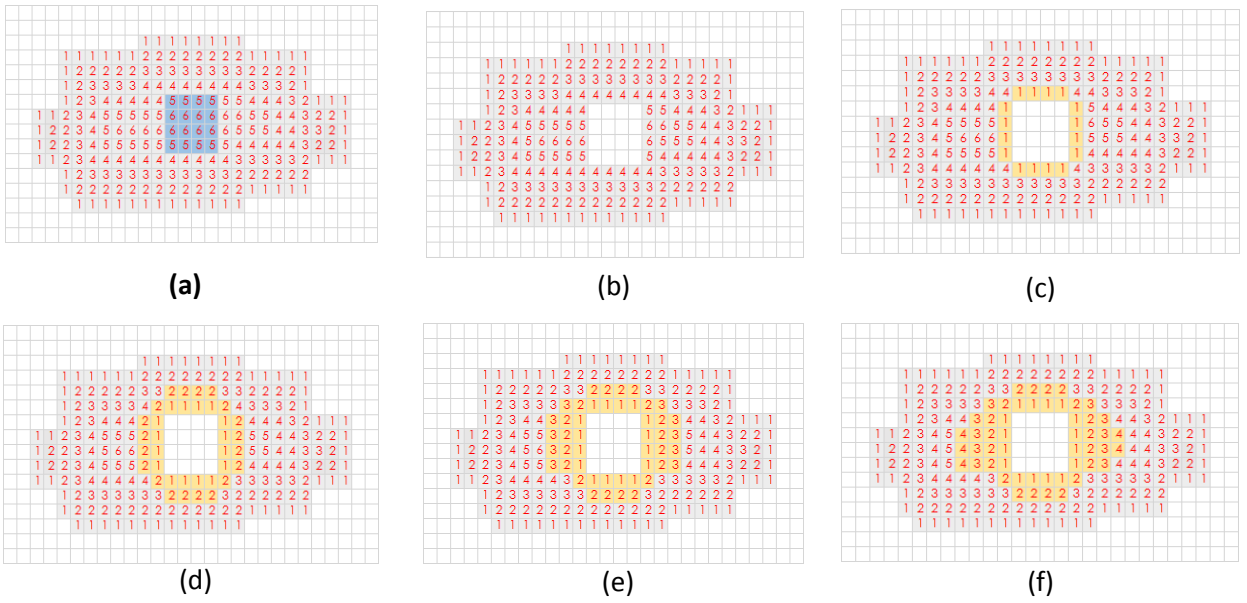


Figure 5:Local Distance Recalculation, (a) a square is drawn, (b) square boundary points are considered as segmentation outline, (c) 4 connected points of the square interior points (now background) are given distance 1, (d), (e), (f) shows the

segmentation outline. This distance calculation is called local as it is maps only pixels nearer than any other segmentation outline. This may be easily handled by parallel distance mapping (***d*₄** or ***d*₈**) taking the square boundary points as the initial set of points. The stopping criterion is newly calculated distances being equal to previous distances as shown in Figure 5. In Figure 6 an actual example of distance recalculation after putting a square is shown.

5.3. Time Complexity:

In this section the complexity of the *Algorithm 2* given above is discussed. The algorithm is run sequentially for each of the denomination in the set D and for each denomination d∈D there is a O(1) computation involved in precalculate_square() function. The inside while loop draws square for a particular denomination d_i and maximum $\alpha_i n/d_i^2$ squares are drawn ,n being number of pixels in image and α_i being fraction of area covered by squares of denomination d_i. Drawing a square of denomination d_i takes O(d_i^2) and an additional O(1) for distance_recalculation() function, as it may be safely assumed that number of points in segmentation outline is O(n) .

(a) (b)

Figure 6 : Distance Mapping after a square fitting (a) d_8 mapping of a segmentation, (b) distance recalculation after putting a square

Hence overall complexity formula is given as,

$$O(|D|).O(1) + \sum_i \alpha_i \mathrm{n}/d_i^2 \left(O\left(d_i^2\right) + O(1)\right)$$

$$= O(|D|) + \sum_i \alpha_i \mathrm{n} \leq O(|D|) + O(n) \text{ , as } \sum_i \alpha_i \leq 1$$

If denomination set is kept contant the algorithm is O(n) .This equation gives the ideal case . In case of squares discarded due to failing variance check it adds an additional O($\sum_i k_i\, d_i^2$) time complexity where k_i number squares of size d_i are discarded. In the result section it is observed that as the number of discarded squares increases the CPU time increases.

6. Results :

In this section square packing results with different distance mapping algorithms, different denomination set and other heuristic methods are compared for their packing density, mean square difference between original and tiled image and time requirement.

6.1. Comparison of three distance metric:

In this section we compare the three different distance metric and their suitablity in fitting squares of different alignment is shown. As mentioned earlier d_8 is suitable for fitting orthogonal squares Figure 7 (b), similarly d_4 for 45^0 axis rotated square (Figure 7(c)) and d_e for square of any alignment (Figure 7(d)).

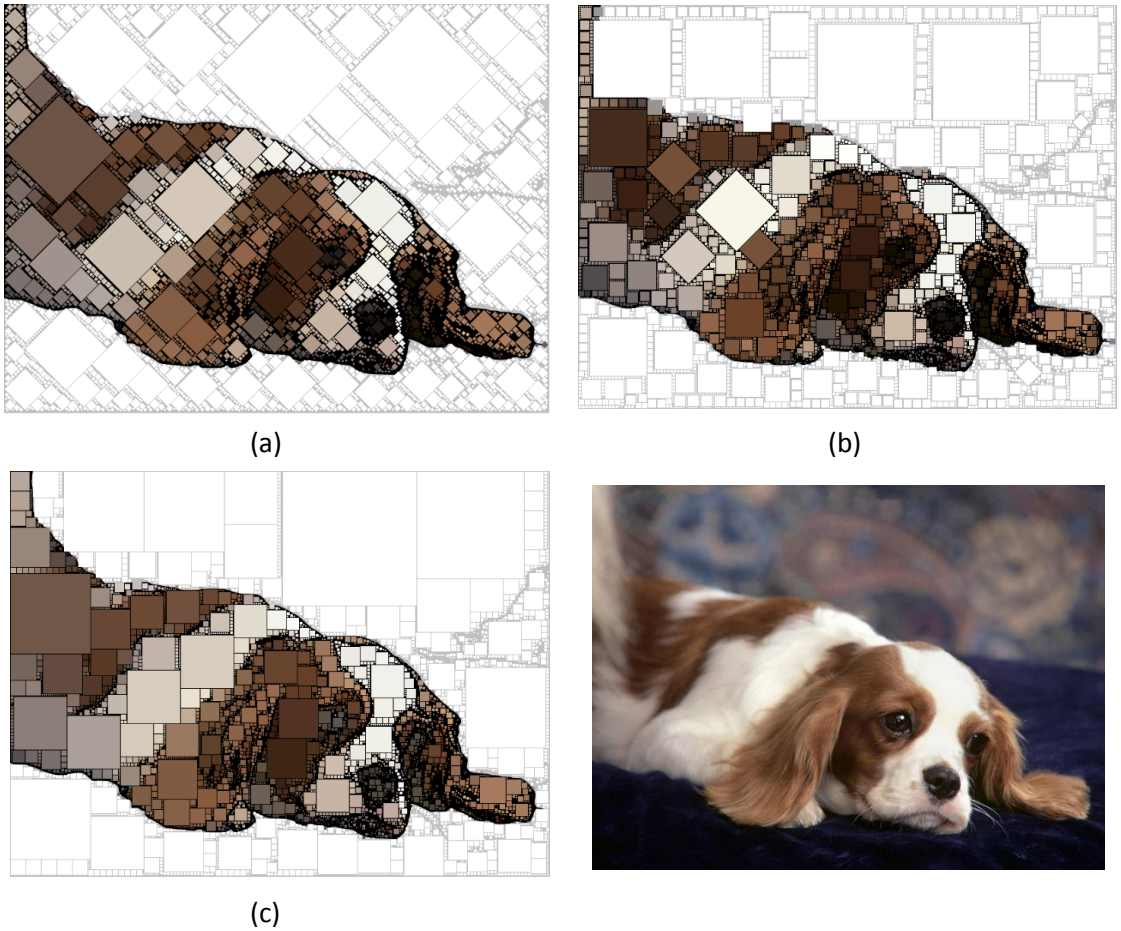
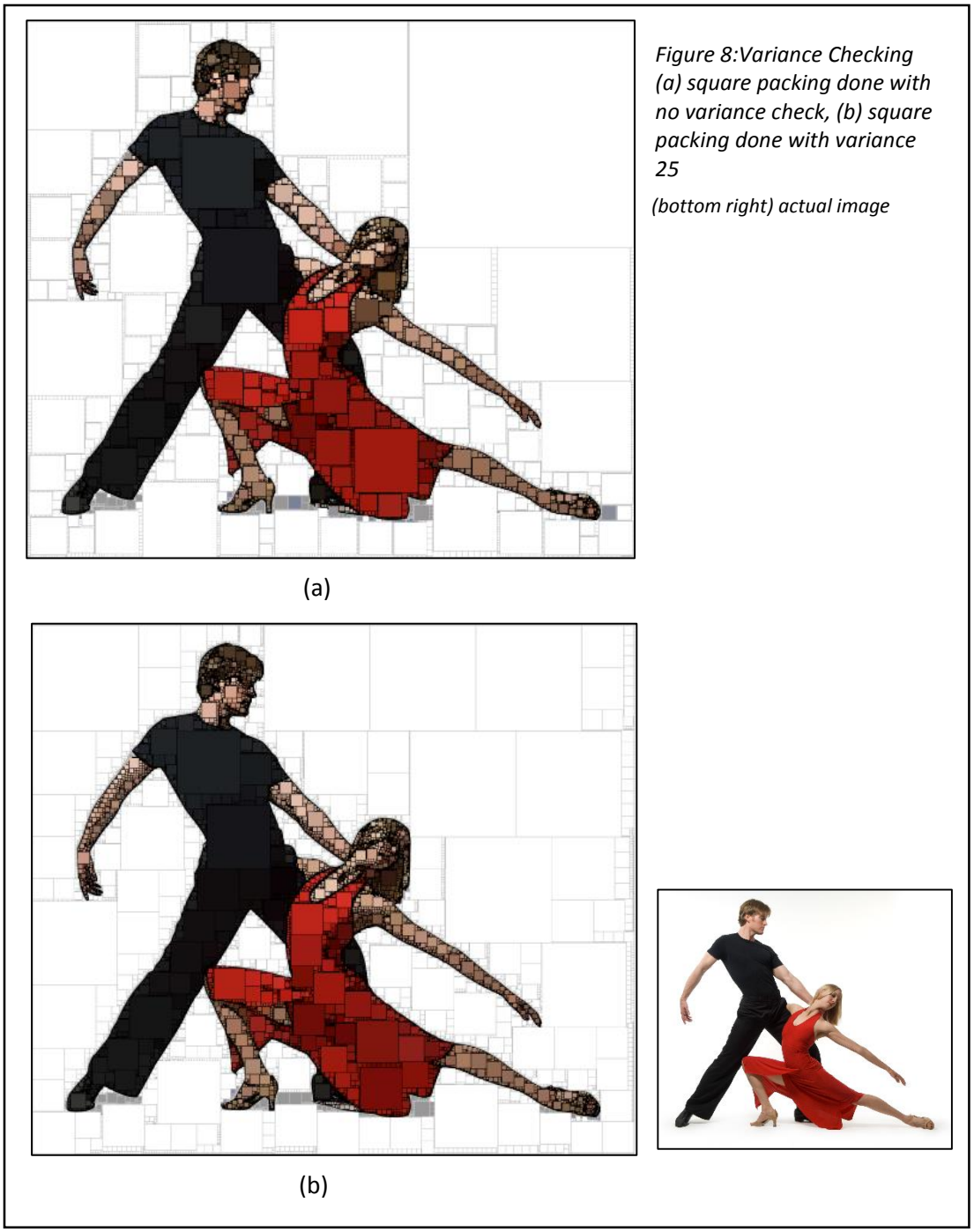


Figure 7: Result comparison of Distance Metrics (a) 45° tilted square fitting d4 distance mapping on d4 distance mapping, (b)square fitting on euclidean distance mapping , (c) axes orthogonal square fitting on d8 distance (bottom right) actual image

6.2. Comparison of algorithm with or without variance check:

In this section we introduce a colour variance checking so that in part of the image with more variance i.e. more detailed part larger squares are substituted by many smaller squares improving the variance of that part.

The result of using different variance threshold is shown in the Figure 8.



(a)

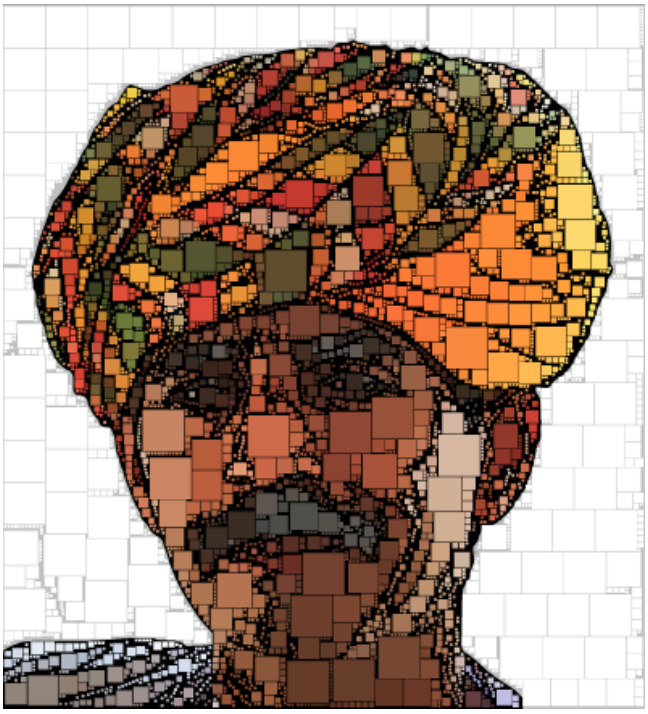
(b)

Figure 8:Variance Checking (a) square packing done with no variance check, (b) square packing done with variance 25

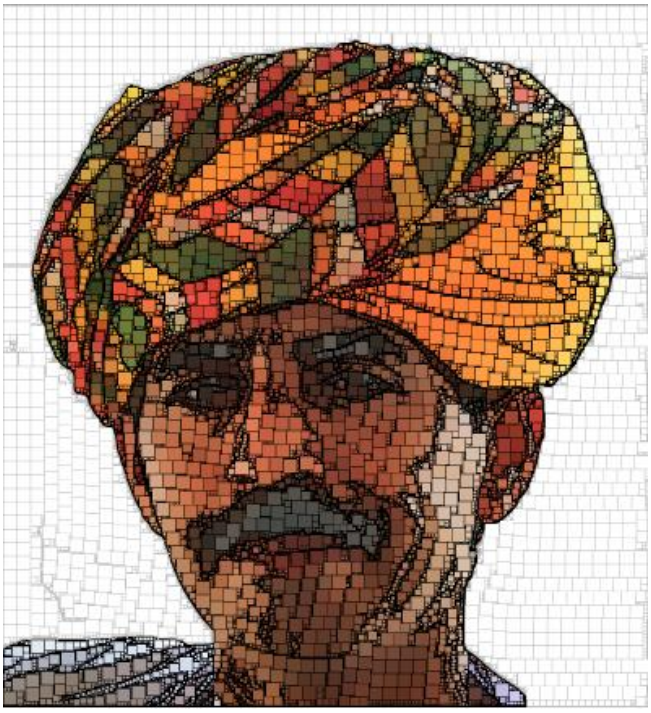
(bottom right) actual image

6.3. Comparison of algorithm different sets denomination:

In this section different denomination sets are used to create different effects. In fig. 9(a) no denomination is provided and at any pixel the largest possible square is fitted and that results in less number of squares.



(a)



(b)

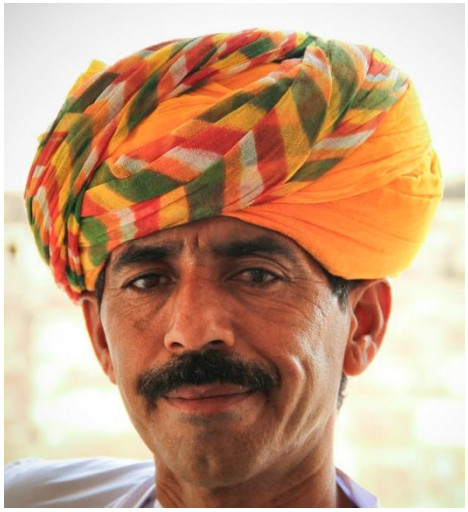
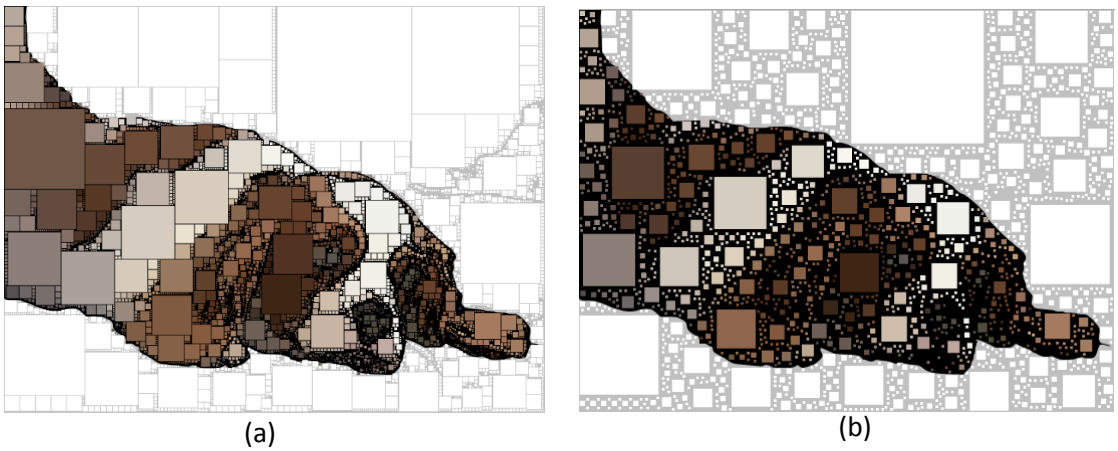


Figure 9: Different denomination sets (a)(30,25,20,16,12,10,5,3,2), (b)(10,8,7,5,3,2) (bottom right) actual image

6.4. Comparison of d_4 and d_8 as local distance recalculation metric:

In this section it is observed that $\mathbf{d_4}$ is better than $\mathbf{d_8}$ in case of local distance recalculation. This observation is shown in



(a) (b)

Figure 10: Local Distance Recalculation Metric, (a) d_4 , (b) d_8

6.5. Numerical Analysis of Result:

The square packing algorithm implementations are done in Java on a Intel(R) Core(TM) i5-2410M CPU @2.30 GHz machine with 4 GB RAM running Windows 7. Here we present numerically all the results using different mapping techniques, denominations sets and colour variance checking as shown above . The mean shift clustering algorithm is implemented in Matlab as it has optimized matrix processing libraries. The clustering algorithm takes a few minutes for each of the cases.

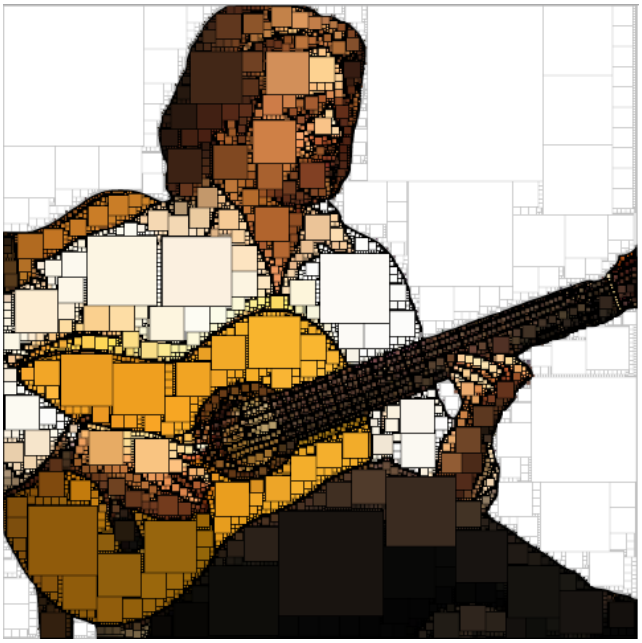
Table 1 : Summary of results, DM=distance metric used in distance mapping, N_r= number of regions after meanshift, D=denomination set, N_s=number of squares after packing,Thr = threshold of colour variance check,ρ = Packing density,DR=Distance recalculation

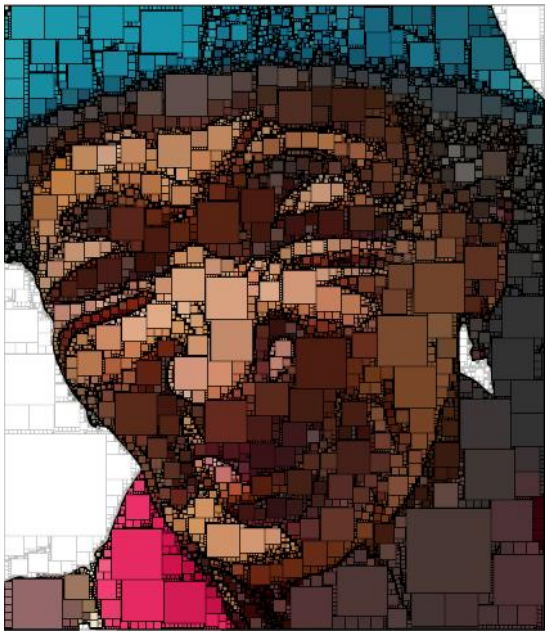
Image Fig. No.	size	DM	N_r	D	N_s	Thr	ρ	Cpu Time
Dog 7(a)	800x600	d_4	17	D1	4210	∞^*	0.8497	8 m16 s
Dog 7(c)	800x600	d_8	17	D1	3900	∞	0.9204	4 m07 s
Dog 7(b)	800x600	d_e	17	D1	6537	∞	0.9020	4 m19 s

Image Fig. No.	size	DM	N_r	D	N_s	Thr	ρ	Cpu Time
Salsa 8(a)	997x1132	d_8	31	D1	5566	∞	0.9517	4 m 6 s
Salsa 8(b)	997x1132	d_8	31	D1	6176	50	0.9126	9 m 59s
Turban 9(a)	896x982	d_8	20	D2	5413	100	0.8775	10m16s
Turban 9(b)	896x982	d_8	20	D3	12082	100	0.8523	17m45s
Dog 10(a)	800x600	d_8 ($DR=d_4$)	17	D1	3900	∞	0.9204	4 m07 s
Dog 10(a)	800x600	d_8 ($DR=d_8$)	17	D1	2153	∞	0.7104	20m15s

[D1=(100,80,60,40,30,25,20,15,12,10,8,5,3,2), D2=(30,25,20,16,12,10,5,3,2), D3=(10,8,7,5,3,2)]

*Thr=∞ means no threshold checking





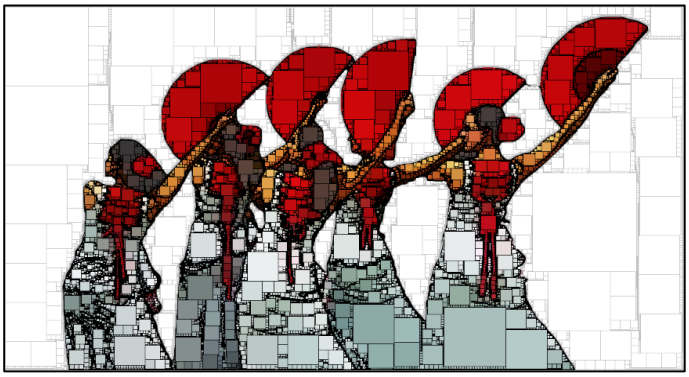


Figure 11 : Some more examples

7. Conclusion:

The detailed analysis of all the results shows that the d_8 distance mapping is most effective solution in achieving highest pacing density and also uses least number of squares.The local distance mapping is best achieved with d_4 as it may be effectively computed in *O(1)* time compared to d_e which takes *O(n)* time.Also d_4 is better metric compared to d_8 as seen in the result section. The time requirement increases as the number of squares discarded due to crossing color variance threshold or overlapping squares increases. The boundaries of segmentation regions are untouched by square boundary to achieve separation between different segments.If the segmentation boundary regions were not left for prominence the packing density would have been improved.Though the whole process is done by algorithms human jugdement is required in deciding parameters to give most visually appealing results. The run time may be further improved by running the square packing algorithm separately in different segments by multithreading, but for that the regions should not be physically connected.

References

[1] R. Verostko, “http://www.verostko.com/algorithm.html”.

[2] “https://en.wikipedia.org/wiki/Algorithmic_art”.

[3] “http://mathworld.wolfram.com/PenroseTiles.html”.

[4] “http://www2.stetson.edu/~efriedma/cirinsqu/”.

[5] R. Harren, “Approximating the orthogonal knapsack problem for hypercubes,” in *International Colloquium on Automata Languages and Programming (ICALP 2006)*, 2006.

[6] K. Jansen and G. Zhang, “Maximizing the total profit of rectangles packed into a,” in *Algorithmica*, 2007, pp. 47, 323–342.

[7] D. Kleitman and M. Krieger, “An optimal bound for two dimensional bin packing,” in *16th Annual IEEE Symposium on Foundations of Computer Science (FOCS 1975)*.

[8] K. Jansen and R. Solis-Oba, “13th International Conference, IPCO 2008 Bertinoro, Italy, May 26-28, 2008 Proceedings,” in *Integer Programming and Combinatorial Optimization*, 2008, pp. 184-198.

[9] M. M. Paulhus, “An Algorithm for Packing Squares,” *Journal of Combinatorial Theory, ,* no. Series A 82,, 1997.

[10] S. De and P. Bhowmick, “Digital Circlism as Algorithmic Art,” in *9th International Symposium on Visual Computing*, Rethymnon, Crete, Greece, 2013.

[11] D. Comaniciu and P. Meer, “Mean Shift: A Robust Approach Toward Feature Space Analysis,” *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 24, no. 5, pp. Page 603-619, 2002.

[12] P.-E. Danielsson, “Euclidean distance mapping,” *Computer Graphics and Image Processing,* vol. 14, no. 3, pp. 227-248, 1980.