University of West Bohemia
Department of Computer Science and Engineering
Univerzitní 8
306 14 Plzeň
Czech Republic

Alternative Representation of Image Information

Josef Kohout

Distribution: public

Technical Report No. DCSE/TR-2009-11
July, 2009

Technical Report No. DCSE/TR-2009-11

Alternative Representation of Image Information

Josef Kohout

Abstract

Information in images (as well as in volume data or video) is typically represented by an array of pixels, where each pixel stores either greyscale luminance or colour components values. Due to the simplicity of this representation, many algorithms from signal processing can be implemented efficiently with ease. For some applications, however, this representation may not be the best one as it is quite space consuming (although the storage requirements can be reduced by using image compression techniques such as JPEG) and is liable to the occurrence of aliasing artefacts. This thesis describes possibilities of alternative representation of image information based on the exploitation of various geometrical data structures such as triangulations (e.g., Delaunay triangulation), etc. It investigates various methods for the evaluation of the significance of pixels and it deals with the problem how to reconstruct image data from significant pixels only. Various methods for the storing of significant pixels in compact forms are also discussed. Methods proposed in this thesis are compared with already existing methods. The thesis also describes the possible extension for video and discusses the options of direct manipulation with images represented by triangulations.

This work was supported by the GA AV of the Czech Republic – project KJB10470701

Copies of this report are available on http://www.kiv.zcu.cz/publications/
or by surface mail on request sent to the following address:

University of West Bohemia
Department of Computer Science and Engineering
Univerzitní 8
306 14 Plzeň
Czech Republic

Background

This report describes results of the research project KJB10470701 (Alternative representation of image information by the use of triangulations) of GA AV of the Czech Republic. The project was being solved in 2007 – 2009 by a team composed of Ing. Josef Kohout, Ph.D. (who was its main researcher), Doc. Dr. Ing. Ivana Koligerová and several undergraduate students supervised either by Josef Kohout or by Ivana Kolingerová. Those students were: Tomáš Janák (interpolations on triangulations), Radek Sýkora (extension of proposed methods for colour images), Petr Puncman and Martin Varga (extension for video) and Josef Vyškovský (direct manipulation with images represented by triangulations).

Important note: as the writing of this report started in 2008 and finished before the end of the abovementioned project, some of its sections may not include the final results.

Basic terminology

In this subsection, we explain various basic terms that are used in the next text of this thesis. Advanced terms will be explained in the text at the place where they firstly appear.

k-Simplex, with $k \leq d$, is the convex combination of k +1 affinely independent points in a point set S in E^d. These points are called vertices of the simplex. In E^1 it is a line segment, in E^2 a triangle and in E^3 a tetrahedron. In this thesis, we use the term simplex also as a synonym to the term node. When we, therefore, speak about a modification of a simplex or an access to a simplex, we mean, actually, the modification of the node that stores information about this simplex or the access to the node data structure.

Convex hull CH(S) of a set of points S is the smallest convex geometrical object (polygon in E^2 and polyhedron in E^3) such that any point from S lies inside the interior of $CH(S)$ or it is one of the vertices of $CH(S)$.

Divide & Conquer denotes a recursive strategy consisting of two stages. In the first one, the divide stage, the input data set is repeatedly subdivided as equally as possible into smaller subsets until each subset is small enough to be solved directly. Afterwards, the solution for each subset is found. In the second stage, the merge stage, solutions of subsets (i.e., subsolutions) are repeatedly merged until the solution for the whole input set is obtained.

Used Shortcuts

The shortcuts commonly used in this thesis are summarized in the following table:

2D	E^2	two-dimensional case, i.e., planar case
3D	E^3	three-dimensional case
CH(S)		convex hull of S
D&C		Divide & Conquer
DT	DT(S)	Delaunay triangulation
MS		Microsoft

Table of Contents

1 Introduction

A digital image, volume data or video (as the latter two can be considered as a set of images) is typically represented by a raster of pixels (an array of pixels), where each pixel stores either greyscale luminance or colour components values. Despite its popularity (mainly due to its simplicity), this representation suffers from several disadvantages. First of all, it is quite space consuming. This is especially true for video; one minute long colour video in the resolution 640×480 consumes approximately 1.3 GB (assuming 25 frames per second). That is why images are commonly transferred and stored in compact forms, such as GIF, PNG and JPEG (MPEG or DivX for video). Unfortunately, these forms are not well-suited for applying further image processing operations directly in the compressed domain. Next problem is that scaling and rotation operations applied on an image in this representation typically introduce some distortion to the image. Despite our best effort, sharp edges present in an image are blurred (or converted into a set of squares) after the image is enlarged.

Besides raster images, there are also vector images that usually contain simple geometrical objects described by their analytical functions or by coordinates of vertices that form them. Vector representations (here aka geometric representations) do not suffer from distortions caused by affine transformations and they have also a potential to be more compact. As data acquisition devices (e.g., digital camera) produces images in raster representation, it is necessary to transform these images into some geometric representation in order to exploit these advantages. However, whilst the transformation from any geometric representation into raster representation is straightforward, for the conversion of digital images from the traditional raster representation into a geometric representation is complex and ambiguous.

If grey-scale images are considered only, the straightforward approach is to think about the pixels of raster image as about 3D points in a space where x and y-coordinates are the rows and columns of the image, and z-coordinate is the appropriate grey level. These points can be connected to form non overlapping polygons, e.g., triangles. There are, indeed, many ways how to do it. Another issue is that it may not be very useful to represent an image with N pixels by a geometric representation (e.g., triangulation) with the same number of vertices. A representation such that it has fewer vertices but it still sufficiently approximates the original image is very often needed to be found. No wonder that these challenges have attracted recently many researches.

Existing methods for the conversion of digital images from the traditional raster representation into a geometric representation can be subdivided into three main categories according to the goal they want to achieve as follows. First, there are methods that produce geometric representations that enhance the quality of further image processing. The representations are not compact as they contain usually as many vertices as the raster. Majority of these methods creates the data dependent triangulation (DDT) where triangle edges match the edges in the image and they differ only in cost functions used to detect an edge and optimisations [Bat04, Su04, Yu01].

In the second category, there are methods that produce compact (i.e., only a subset of vertices is kept) but highly imprecise representations. They find its use in applications of non-photorealistic rendering where details are unwanted because they make an understanding of the information presented by the image more difficult. A typical application of such representations is described in [Gru05]. From existing methods that belong to this category, let us describe two interesting.

Prasad et al. [Pra06] proposed a technique that starts with the detection of edges in the input image using the Canny operator. The detected edges are used as constraints for the constrained Delaunay triangulation that is afterwards constructed. For every constructed triangle one colour computed as the average of colours of pixels covered by the triangle is assigned. Adjacent triangles with similar colours are merged together forming a polygon for which a common colour is chosen. The process results in the polygonal representation of the image.

Kreylos et al. [Kre01] describes an interesting approach that starts with the Delaunay triangulation of a randomly chosen subset of vertices that is successively improved by choosing different vertices governed by a simulated annealing algorithm. A drawback of their approach is that the final triangulation contains a lot of long and narrow triangles that may be difficult to efficiently encode. The approach was later exploited by Cooper et al. [Coo05] for the surface reconstruction from a set of images. Instead of picking a random subset for the initial approximation, they, however, choose detected important points (typically, corners and edges).

The last category consists of methods that attempt to balance the compactness and the quality of the produced geometric representations that, if efficiently encoded, are suitable for the storing of digital photos. These representations are very often adaptive triangulations that differ in the way how they were obtained. In general, we can identify two basic strategies how to create an adaptive triangulation. The first one generates an adaptive triangular mesh by starting with two triangles covering the whole image area and then successively splitting them in order to reduce the approximation error. Alternatively, the algorithm can start with a fine mesh and successively make it coarser until the approximation error is above the desired tolerance. The question is which triangle should be split or which vertex should be removed in the next step and that it is not a simple task is demonstrated by two straightforward approaches described in [Gev97] and [Cia97] that either do not preserve well sharp edges in images or produce meshes with many vertices. Let us describe some more sophisticated approaches.

Starting with two initial triangles and their corresponding approximated image, Rila et al [Ril98] successively construct the Delaunay triangulation as follows. A vertex, in which the approximation is the poorest, is inserted into the triangulation, which results in the construction of new triangles. These triangles are interpolated, i.e., a new approximation is obtained, and the next point to be inserted is found. The process stops when the required quality of the approximation is reached. The authors also describe a technique for the storing of the created mesh. As the Delaunay triangulation of a set of points is unique, it is necessary to store just vertex positions and their grey levels. An array of N bits such that it contains 1 at positions appropriate to the vertices of the constructed triangulation and 0 elsewhere is constructed and compressed using a RLE (Run Length Encoding) approach. The grey levels are encoded using a fixed-length uniform quantizer of 5 bits.

García et al [Gar99] choose a predefined number of pixels from image by applying a non-iterative adaptive sampling technique, which detects pixels on edges present in the image, and triangulate the corresponding points of these pixels using the Delaunay triangulation. Afterwards, triangles are further subdivided as long as the error of the approximation does not drop below some threshold. Although the authors were able to achieve better results (in the compression ratio as well as in the quality of the representation) than the authors of straightforward approaches, the results are, in our opinion, still far from being perfect – see Figure 1.1.

In the approach described by Galic et al [Gal05], a vertex with the poorest approximation is found using the same criteria as Rila et al. [Ril98] in every step of their algorithm and the triangle containing this vertex is split into two new triangles by the height on its hypotenuse. The centre of the hypotenuse becomes an additional vertex of the representation. The advantage of this hierarchical splitting process is that it forms a binary tree structure that can be

efficiently stored using just one bit per node. For the encoding of grey levels, the authors use the Huffman compression. In their paper, they also discussed various interpolation techniques and finally they decided to use edge-enhancing diffusion interpolation for their experiments instead of commonly used piecewise linear interpolation.

Figure 1.1: *Lena (512x512 pixels) represented by an adaptive triangular mesh of 5807 vertices constructed by the approach by García et al. Image was adopted from [Gar99].*

More recently, Demaret et al. [Dem04] proposed an algorithm that computes the Delaunay triangulation of all vertices and after that it successively decimates this triangulation by removing the least significant vertex in every step. A vertex is considered to be the least significant, if its removal leads to the approximation of the original image with the smallest mean square error (MSE). The authors were able to achieve the compression ratio comparable with JPEG and the same or, especially, for higher compression ratios, even better quality of the image representation – see Figure 1.2. On the other hand the proposed algorithm consumes a lot of time.

Figure 1.2: *Lena (512x512 pixels) represented by an adaptive triangular mesh constructed by the approach by Demaret et al. The compression rate is about 53:1. Image was adopted from [Dem06].*

A hybrid approach is described in [Part03]. The original image is first segmented using an unsupervised segmentation method for colour-texture regions. Following polygonal approximation of created regions causes the degradation of region boundaries. The triangulation is then applied to polygons and either all short edges, or all small triangles are filtered out from the triangular mesh (CDT or greed approach is used). It results in new smaller regions – see Figure 1.3. Pixels in every region are then independently encoded with a code similar to JPEG (different quantization can be applied for different regions). The proposed method is better than JPEG representation but it offers only a limited set of advantages of the geometric representation in comparison with previously described approaches.

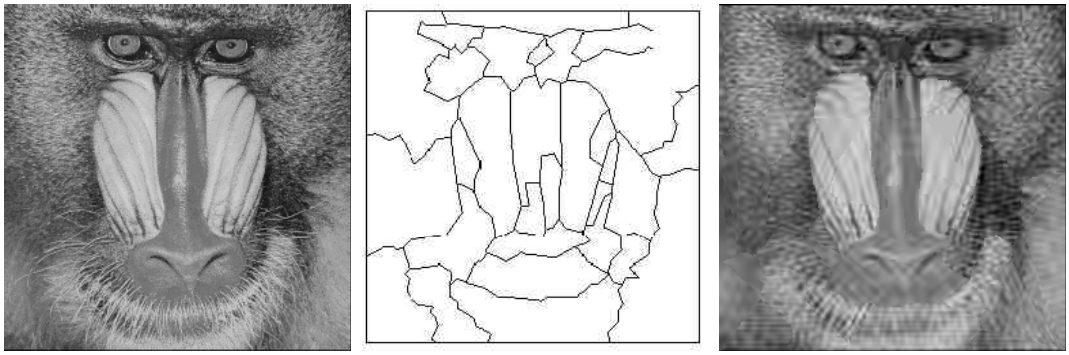


Figure 1.3: *Baboon (256x256 pixels) compressed by the approach by Partyk et al. The compression rate is about 27:1. Image was adopted from [Par03].*

A digital video (or 3D image information) is a stream of similar digital images. Usually, these images are denoted as frames. Videos are commonly transferred and stored in compact form through well-known representations, such as MPEG or DivX. These representations encode image frames by the JPEG compression technique combined with the strategy to use previous frame information in order to reduce the amount of information the current frame requires.

There is not so much done in the field of geometric representation of digital video. Yaoping et al. [Yao98] proposed quite a straightforward approach that replaces the traditional JPEG representation of frames by the alternative geometric representation by the Delaunay triangulation. This triangulation is constructed by the successive application of split-merge scheme, i.e., it is a combination of approaches described in [Gev97] and [Cia97]. The authors show that, for very low bit-rate transmissions, their representation offers higher quality of decoded frames than the standard approaches.

The main goal of this work is to represent grey-scale and colour images and videos by a geometric representation that approximates the original data in an acceptable quality, yet it is more compact than the traditional raster representation. This is achieved by keeping only a small subset of the most significant pixels. Assuming that we deal with 8-bits grey-scale image, this subset may contain no more than 20% of original pixels because positions of selected pixels must be, unlike in the raster representations, also retained (we suppose that the position can be stored using a pair of two bytes long integers). The computation of the smallest subset of pixels that represents the image in the desired quality (i.e., with the given error) is NP-hard problem. The investigation of every subset of 1 000 pixels for an image of 512×512 pixels would take several millions years[1]. Some heuristics is, therefore, necessary.

Let us suppose we already have an algorithm that can create a subset of pixels such that all other pixels of the image can be reconstructed from it by some interpolation in a requested quality. The problem is that existing interpolations of scattered points (pixels in our case) are usually too slow to be used in interactive applications because they must investigate the relationship between every point to be reconstructed and every point from the input set. For an instance, the interpolation of 4 000 pixels selected from an image 512×512 by the approach proposed by Uhlíř et al. [Uhl05], which is based on radial basis functions (RBF) used in a sliding window, takes about 90 seconds on a P4 computer with 2GB of RAM. In order to achieve even better results, it is necessary to organize the points into some structure.

The organization of this report is as follows. In the following section, we describe the most popular triangulations and their constructions. Section 3 proposes various heuristics selection of most significant pixels (let us note that the triangulation and selection are two mutually dependent tasks) for grey-scale images. The performed experiments and their results are given in Section 4. For storing purposes, the triangulations can be further compressed. Various triangulation compression strategies are described in Section 5 and they are compared in Section 6. Section 7 deals with the interpolation of triangles. The extensions for colour images and videos are described in Section 8 and Section 9. Direct manipulations with images represented in the proposed geometric format, e.g., smoothing, convolution, etc. is given in Section 10. The report is concluded in Section 11, which also discusses the future work.

[1] There is $\binom{262144}{1000} = \frac{262144!}{261144!\cdot 1000!} = \frac{262144\cdot 262143\cdots 261145}{1000\cdot 999\cdots 1} > \frac{261144^{1000}}{1000^{1000}} > 261^{1000} \cong 10^{2416}$ different sets.

2 Triangulations

Given a point set S in E^2, the triangulation $T(S)$ of this set is a set of triangles such that:

- The point $p \in E^2$ is a vertex of a triangle from $T(S)$ if and only if p belongs to S; i.e., the vertices of the triangles are some points from the input set.
- The intersection of two triangles is either empty or it is a shared face, a shared edge, or a shared vertex.
- The set $T(S)$ is maximal: there is no triangle that can be added into $T(S)$ without violating previous rules; i.e., union of triangles and convex polygon formed by a convex hull $CH(S)$ is the same object.

One advantage of triangulation is that it divides the image space, which allows an easy detection of pixels that should be taken into account for the interpolation and those that should not. Let us note that the bilinear interpolation on triangulations is implemented in every graphics adapter, so it is possible to reconstruct images from triangulations in real-time.

It is clear that one set of points can be triangulated in various ways. There is also no doubt that the interpolation of two different triangulations of the same set may produce different results. Therefore, the choice of triangulation is as important issue as the selection of subset of significant pixels. In this section, we describe Delaunay, constrained Delaunay, regular and data dependent triangulations that are most suitable for the purpose of image representation. Furthermore, we describe methods of their constructions.

2.1 Delaunay Triangulation

Delaunay triangulation was proposed by a Russian scientist Boris N. Delone [Del34a, Del34b]. However, as his original papers are not written in English and their translations are usually rather complex, we would recommend Radke's [Rad99] or de Berg's [Ber97] texts for details about Delaunay triangulation.

Delaunay triangulation $DT(S)$ of a set of points S in E^2 is a triangulation such that the circum-circle of any triangle does not contain any other point of S in its interior. In the next text, this criterion is also called the circum-circle criterion.

There is also an alternative definition of the Delaunay triangulation: the DT is a dual of the Voronoi diagram $Vor(S)$, which is a set of points having the same distance from at least two points from S and, moreover, there is no other point from S with a smaller distance. The mathematical expression of the $Vor(S)$ can be written as:

$$Vor(S) = \{x \in E^d : \forall p_i \in S, \exists p_j \in S; i \neq j : |p_i x| = |p_j x| \wedge \neg \exists p_k \in S; k \neq i; k \neq j : |p_k x| \leq |p_i x|\}$$

Figure 2.1 shows the mutual relationship of the $Vor(S)$ and the $DT(S)$.

The basic properties of the $DT(S)$ are as follows [God97]:

- In the worst case, it can be computed in $O(N \cdot \log N)$. However, algorithms with $O(N)$ expected time also exist.
- It maximizes the minimal angle and, therefore, the Delaunay triangulation contains the most equiangular triangles of all triangulations (i.e., it limits the number of too narrow triangles that may cause problems in further processing, e.g., in the interpolation).
- If no four points lie on a common circum-circle and no three points lie on a common line, then the $DT(S)$ is unique. Let us note that four points lying in the vertices of an

empty square in E^2 have a common circle and two possible configurations of their triangulation. As pixels lie in a regular grid, the Delaunay triangulation of a subset of pixels is typically ambiguous. However, if a small random perturbation is applied to pixel coordinates, it is very likely to get a unique triangulation, which opens a new option for the encoding of the computed triangulation – see Section 5.

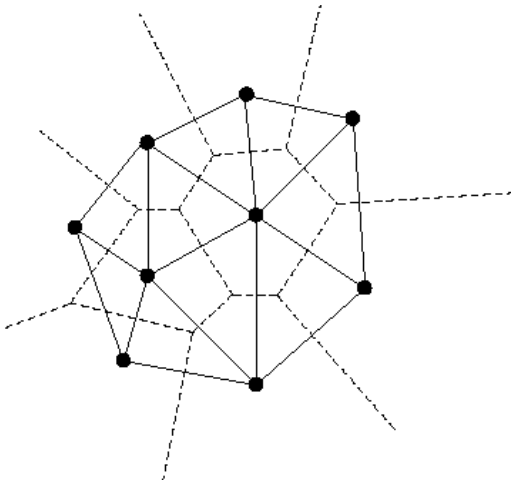


Figure 2.1: *The Delaunay triangulation (solid lines) and the Voronoi diagram (dashed lines) of the same set of points (big black dots)*

Due to these good properties, Delaunay triangulation is used in many areas, such as terrain modeling (GIS) [Gon02], scientific data visualization [Oku96, Oku97, Wal00, Att01] and interpolation [Par03], robotics, pattern recognition [Pra00, Xia02], meshing for finite element methods (FEM) [Béc02, Nis01], natural sciences [Mul03, Ada03], computer graphics and multimedia [Ost99, Tek00], etc.

Many algorithms for construction of the Delaunay triangulation of the given point set exist. Some of them exploit the duality and construct the Delaunay triangulation from the Voronoi diagram, whilst others compute the Delaunay triangulation directly. We classify direct algorithms into several categories: local improvement, incremental construction, incremental insertion, higher dimension embedding and divide & conquer [Koh05]. Except for incremental insertion algorithms, which are also known as online, all points must be known before the triangulation process starts. As the selection of the optimal subset of pixels and their triangulation are not two separate steps (see Section 3), points are not known in an advance and, therefore, an incremental insertion algorithm is the only option we have.

2.1.1 Incremental Insertion with Local Transformations

Starting with an initial Delaunay triangulation, e.g., an auxiliary simplex that contains all points in its interior, the algorithm inserts the points in the input S into existing Delaunay triangulation one at a time. As long as we do not consider time requirements, the order of the insertion is not important. The points do not need to be known in advance (although their range of coordinates is needed). If the algorithm uses a randomized order of insertion, it becomes almost insensitive to the type of points distributions.

The insertion consists of three phases: the *location* where a simplex containing the point to be inserted has to be quickly found followed by the *subdivision* of this simplex and by the *legalization* where the circum-sphere criterion is applied and if it is necessary, the local improve-

ment techniques are used to restore the Delaunay triangulation. The algorithm written in pseudocode is given in Figure 2.2.

Points already present in the triangulation can be also successively removed. The deletion consists also of three phases: the *location* (the same as for the insertion) is followed by the *cavity construction* where all simplices sharing the given point are removed from the triangulation and by the *cavity retriangulation* where new Delaunay triangles are constructed to fill the created cavity. Let us describe all steps in detail.

```
Input: A set S = { p0, p1, ..., pN-1 } of N points in E2

for r := 0 to m - 1 do begin
   Locate the simplex S0 ∈ DT(S) containing pr;

   Subdivide S0;  //in the case where pr lies on the shared edge or face
                  //then subdivide also the appropriate adjacent simplices.

   //Legalize all new simplices
   while there exist an unchecked face F do
      if the face F violates DT criterion
             then perform local transformation;
end;
```

Figure 2.2: *Construction of the DT(S) by incremental insertion with local transformations.*

2.1.1.1 Initialization

Let us have the input set *S* of *N* points. An auxiliary simplex large enough to hold all these points inside its interior is constructed. We prefer this large simplex to the convex hull (see Section 3) because it is easier and, according to our experience, more stable. One problem with this approach is how to choose the vertices of this simplex. If they are not far enough away, they may influence the empty circum-sphere tests, which may lead to the non-convex boundary of the resulting Delaunay triangulation. On the other hand, if the vertices are "too far away", it may lead to numerical instability of the algorithm.

Therefore, in our algorithm, the vertices have coordinates (*K*, 0), (0, *K*), (-*K*, -*K*) for the version in E^2 and (*K*, 0, 0), (0, *K*, 0), (0, 0, *K*), (-*K*, -*K*, -*K*) for the version in E^3. The value *K* is equal to the multiple of the size of the bounding box of points – see Figure 2.3. More detailed description is given by Žalik and Kolingerová in [Žal03].

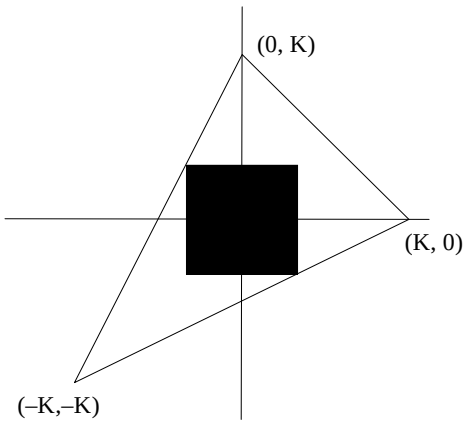


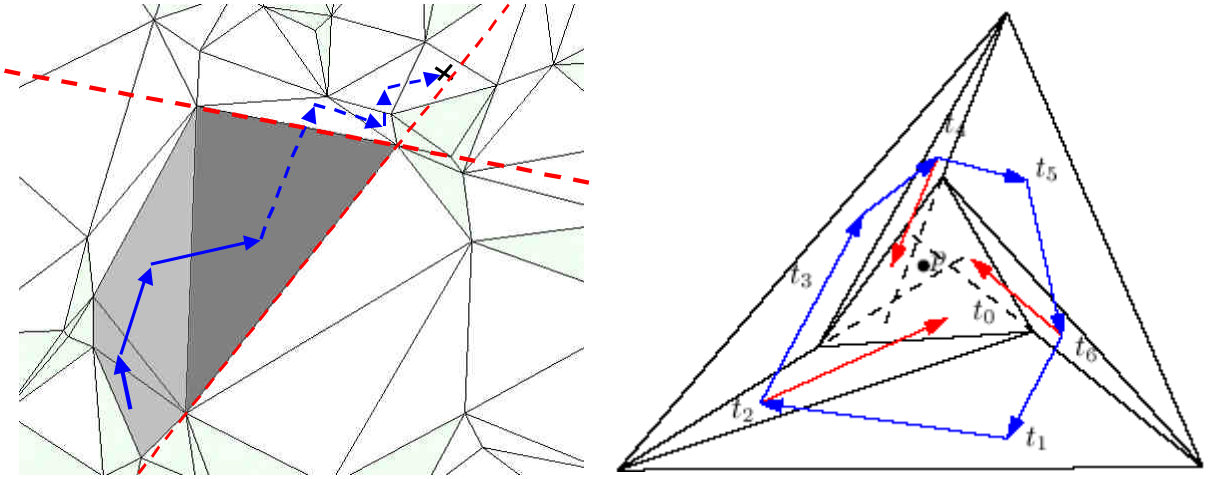
Figure 2.3: *The selection of the auxiliary simplex in E^2. The black rectangle is the bounding box.*

2.1.1.2 Location

In the location part, it is required to find a simplex that contains the given point. It can be done either with use of some hierarchical structures or without them by some walking technique. Walking techniques are based on the searching of simplex to be subdivided directly in the Delaunay triangulation. Therefore, the location can take $O(N)$ time in the worst-case. Fortunately, the worst-case scenario is not very probable and the location is, usually, performed in $O(\sqrt{N})$ expected time. Let us note that under special circumstances expected time $O(N^{1/3})$ can be reached [Žal03]. Although walking approach is a bit slower than the approach with a hierarchical structure, its big advantages is that it needs no additional memory. Different walking techniques are presented in [Dev01].

Let us describe *visibility* walk in E^2. Starting from an arbitrary triangle, the algorithm traverses through the triangulation testing the mutual position of visited triangles and the given input point until the triangle containing this point is found. For each visited triangle, it is necessary to detect an edge such that the line supporting this edge separates the triangle from the input point, which can be reduced to a single orientation test. If there is no such edge, the triangle contains the point in its interior. Otherwise, the search continues with the neighbouring triangle sharing the detected common edge. Figure 2.4a shows an example of walk.

Unfortunately, for non-Delaunay triangulations, the walk we have just described may fall into a cycle as illustrated in Figure 2.4b. As the constrained Delaunay triangulations (i.e., with some prescribed edges – will be discussed in further text), which are important in practice, are also non-Delaunay, a little bit of randomness has to be introduced into the algorithm in order to avoid infinite loops. Instead of starting the detection with the first edge of the given triangle, the algorithm starts with randomly picked edge. This ensures that, if the walk enters a cycle in the triangulation, it cannot loop in this cycle forever. Another small improvement is to remember, for each visited triangle, the edge that was just crossed by the walk and do not test this edge twice. The visibility walk algorithm with these two improvements is called *remembering stochastic walk.*



a) the path of visibility walk, the dark gray triangle is currently being tested, light gray triangles were visited in previous steps

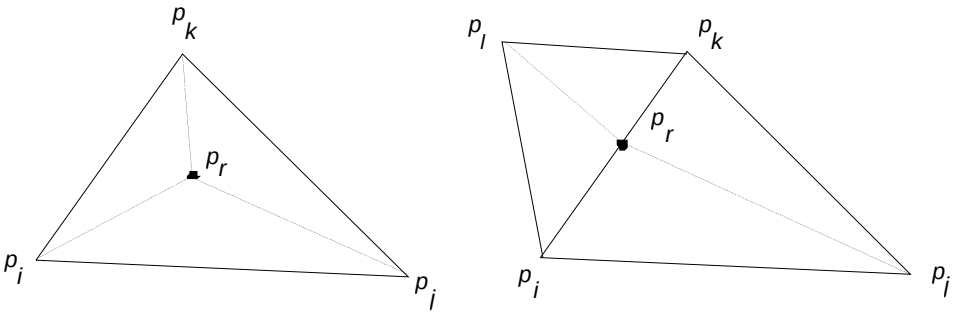
b) an infinite cycle for the visibility walk [Dev01]

Figure 2.4: *The visibility walk algorithm.*

There are other possibilities for quick location. Very popular is Directed Acyclic Graph (DAG) [Ber97], the structure that stores the history of changes. Each inner node of the DAG stores one simplex that existed in some previous triangulation and the current triangulation is stored in the leaves. Time $O(\log N)$ for location is ensured. In the effort to reduce memory use, Devillers in [Dev98] suggests a hierarchical structure similar to the DAG. It consists of several connected levels; each level contains a random sample of the level below. Other possibilities include a use of quadtrees or bucketing techniques. Various techniques for location are compared in [Žal03].

2.1.1.3 Subdivision

Let us suppose we have successfully found the triangle p_i, p_j, p_k containing the point p_r to be inserted. There are several mutual positions of this point and the located simplex. The simplest possible configuration is that the point lies strictly inside the simplex. In this case, all vertices of the located simplex are connected with the point by an edge and the simplex is subdivided into three new simplices (see Figure 2.5a).



a) the point to be inserted lies strictly inside b) the point to be inserted lies on an edge

Figure 2.5: *Subdivision in E^2.*

Slightly more complicated situation occurs when the point to be inserted lies on an edge. It is then necessary to subdivide not only the located simplex but also the adjacent simplex that shares this edge. It results in four new triangles (see Figure 2.5b).

2.1.1.4 Legalization

After the subdivision, we have a new triangulation. However, it may not be the Delaunay one. Therefore, all outer edges of currently created simplices have to be tested whether they do not violate the empty circum-sphere criterion, i.e., whether the far point of the simplex adjacent to the new one does not lie inside the circum-sphere of this new simplex. If the condition is not fulfilled, the triangulation has to be changed by applying the local transformations. The transformation in E^2, which is shown in Figure 2.6, is simple: the edge is just swapped.

After that, indeed, we have new outer edges (or faces) that have to be tested. Figure 2.7 shows an example of the propagation of the local transformations in E^2. The located triangle is subdivided into three new triangles (Figure 2.7a– dotted line). Then, the circum-circle criterion is tested on all these new triangles. The test for the triangle T_1 fails because the far point p_1 of the adjacent triangle lies in the circum-circle of the triangle T_1. The shared edge is flipped. As the circum-circle of the just created triangle T_2 is not empty, the flipping has to continue – see Figure 2.7b. Finally, the Delaunay triangulation is achieved (Figure 2.7c).

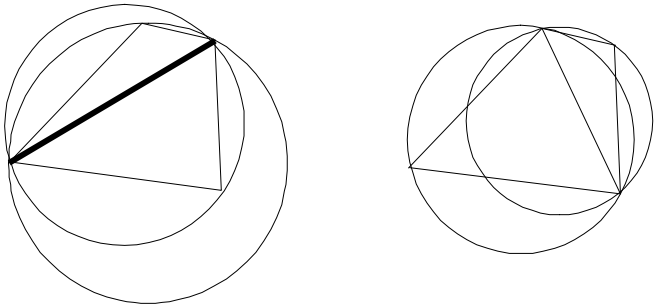
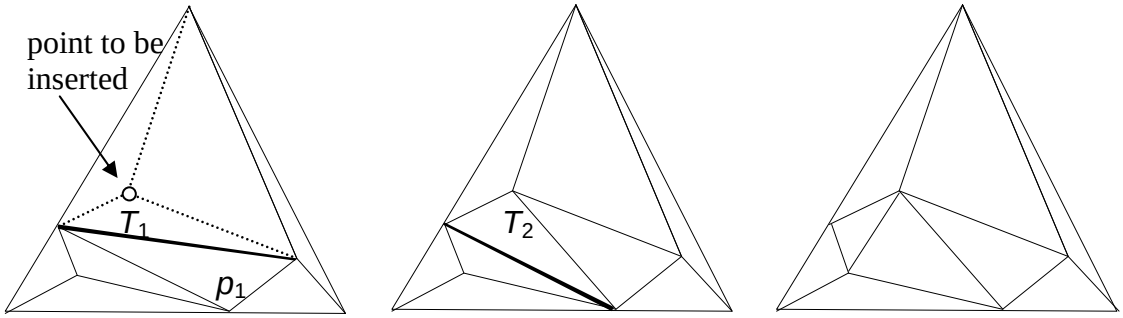


Figure 2.6: *Local transformations in E^2. The edge is swapped.*



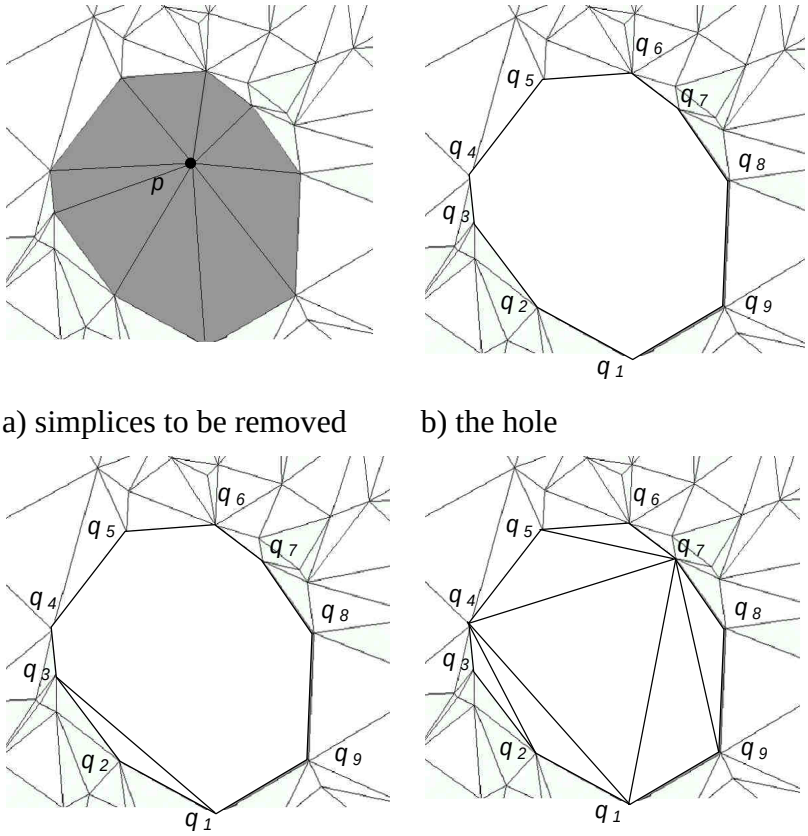
a) subdivision b) propagation of flips c) the resulting triangulation

Figure 2.7: *The incremental insertion in E^2. Edges that should be flipped are bold.*

2.1.1.5 Cavity Construction & Retriangulation

In two dimensions, the deletion of the point p means that m triangles must be removed from the triangulation and $m - 2$ new Delaunay triangles must be created to fill the hole – see Figure 2.8. Although m may be equal to the number of points in the triangulation, it is well known that the expected value of m is 6 without any assumption on the point distribution.

Devillers [Dev99] proposed an efficient algorithm (it requires $O(m \cdot \log m)$ for the retriangulation of hole is based on successive cutting of ears of this hole. For each triple of topologically consecutive vertices q_i, q_{i+1}, q_{i+2} along the boundary of the hole, i.e., for a candidate for the triangle, a weight computed as a function of coordinates q_i, q_{i+1}, q_{i+2} and p is assigned. All candidates are put into a priority queue ordered by their weights. After that an iterative filling process starts. In every step of this process, a candidate at the head of the queue is taken and the corresponding triangle is constructed. The candidates q_{i-1}, q_i, q_{i+1} and q_{i+1}, q_{i+2}, q_{i+3} that overlap the newly constructed triangle are changed to q_{i-1}, q_i, q_{i+2} and q_i, q_{i+2}, q_{i+3} and their weights are recalculated. The process stops when the hole is filled. An example of retriangulation of hole is shown in Figure 2.8.



a) simplices to be removed b) the hole

c) the first ear to be tested d) Delaunay retriangulation

Figure 2.8: *The deletion of the point p from the Delaunay triangulation.*

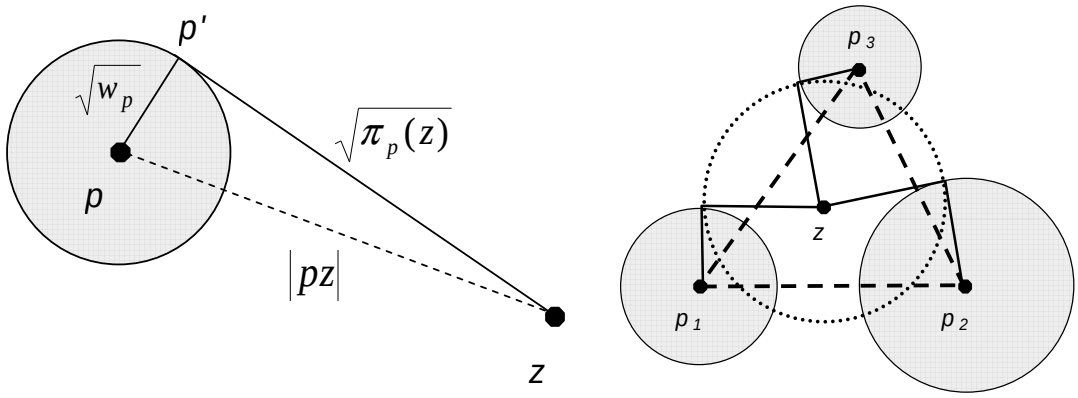
2.1.1.6 Finalization

When the construction has been finished, all simplices having at least one vertex of the big auxiliary simplex are removed from the triangulation.

2.2 Regular (Weighted) Triangulation

Regular triangulations [Ede92, Fac95] are a generalization of Delaunay triangulations offering an extra degree of freedom by introducing weights for points. Given a point set S in E^d, a real valued weight w_p is assigned to every point p from the set. Let us note that the weighted point can be interpreted as a sphere with center p and radius $\sqrt{w_p}$. For each weighted point p, we define so-called *power distance* from a not weighted point $z \in E^d$ to the point p as $\pi_p(z) = |pz|^2 - w_p$, where $|pz|$ is Euclidian distance between points p and z. The geometrical meaning of the power distance is shown in Figure 2.9a.

For any simplex, it is possible to find a point z such that the power distances from this point to every point of the simplex are the same – see Figure 2.9b. A weight equal to the square of the computed value of power distance is assigned to the point z. The weighted point z is called the *orthogonal center* of the simplex and the sphere with radius $\sqrt{w_z}$ centered at z is the *ortho-sphere* of the simplex. Let us note that if the weights of points of this simplex are zero, then the orthosphere and the circum-sphere of the simplex are identical.



a) the power distance from z to p b) the orthogonal center of simplex p_1,p_2,p_3

Figure 2.9: *The geometrical meaning of power distance and orthogonal center in E^2.*

A triangulation is regular only if all simplices are locally regular. A simplex p_1, p_2, p_3 or, in the case of E^3, p_1, p_2, p_3, p_4 is locally regular if the power distance from any point $q \in S - \{p_1, p_2, p_3, p_4\}$ to the orthogonal center of the simplex is larger than the weight w_q assigned to this point q, i.e., $\pi_z(q) > w_q$. It is clear that the method of incremental insertion with local transformations described in the previous text can be used also for the construction of regular triangulation. All that is needed is it to supersede the Delaunay empty circum-sphere condition by the condition of regularity. Points are successively inserted into the existing regular triangulation and, as in the Delaunay triangulation, if a set of adjacent simplices violates the condition of regularity, local transformations have to be applied. Figure 2.10 shows an example of local transformation in E^2. The edge shared by two adjacent triangles is invalid, i.e., the triangles are not regular, and, therefore, it is swapped.

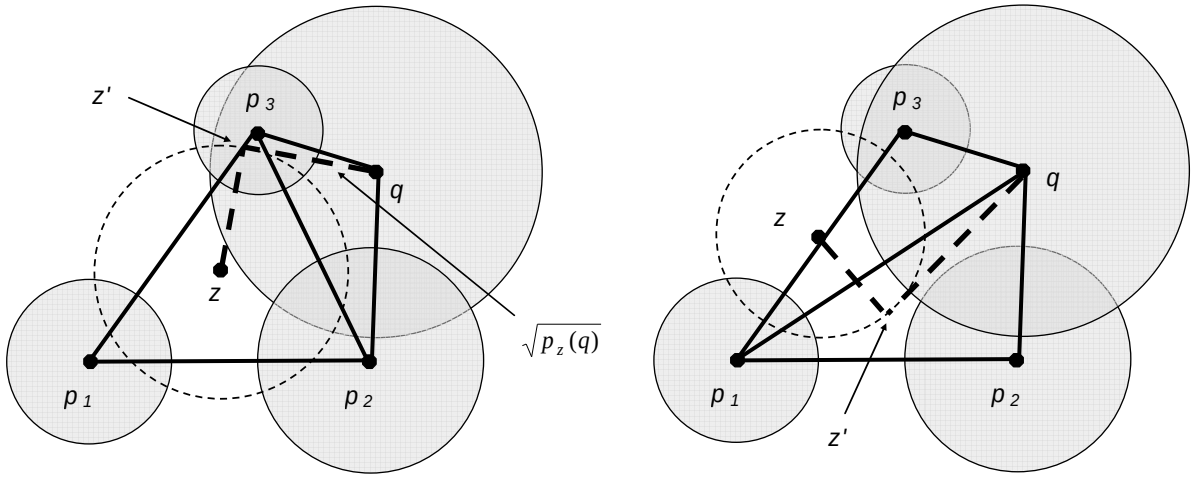


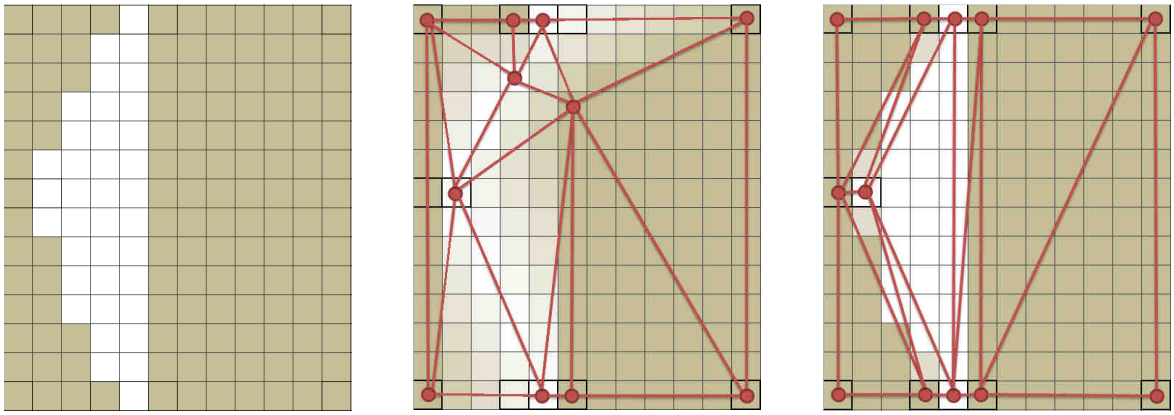
Figure 2.10: *Local transformations in E^2. The edge is swapped.*

If the geometrical meaning of power distance and orthogonal center is taken into account, we can reformulate the condition of regularity as follows. A simplex is regular, if for any point q from S (except for points in the vertices of the simplex) the point z' of contact of tangent to the orthosphere of the given simplex going through the point q does not lie inside the sphere with radius $\sqrt{w_q}$ centered at the point q – see Figure 2.10. This means that to decide whether an

edge is invalid, i.e., whether a local transformation must be applied or not, we need to test the mutual position of some sphere and point. It is exactly the same test as the one used in the Delaunay triangulation, only spheres and points to be checked are different.

Similarly to the Delaunay triangulation, the regular triangulation is unique and, therefore, the topology does not need to be stored as long as we preserve weights for selected pixels. Without any doubt, the storing of weights negatively influences the compactness of the representation. On the other hand, the regular triangulation, if points are properly weighted, can better represent image features and, therefore, fewer points are required – see Figure 2.11.

There is also another option for the storing of regular triangulations. Kim et al. [Kim99] showed that the Delaunay triangulation of the given terrain data set (grey-scale images are close to terrains) is very similar to other common triangulations of the same data set. Therefore, instead of keeping all weights, it is possible to store coordinates of vertices (and the associated data) and save those edges that are not present in the Delaunay triangulation. Considering that weights are real numbers (hence their encoding typically requires 4 bytes per one weight), this strategy can produce more compact result.



a) the original image

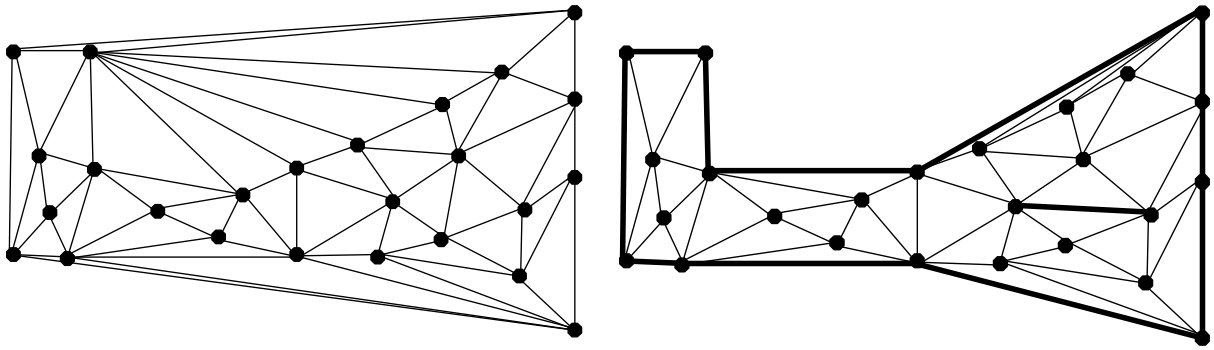
b) the image reconstructed from the DT

c) the image reconstructed from the regular triangulation

Figure 2.11: *The reconstruction of image from the Delaunay triangulation and the regular triangulation of the same number of points.*

2.3 Constrained Delaunay Triangulation

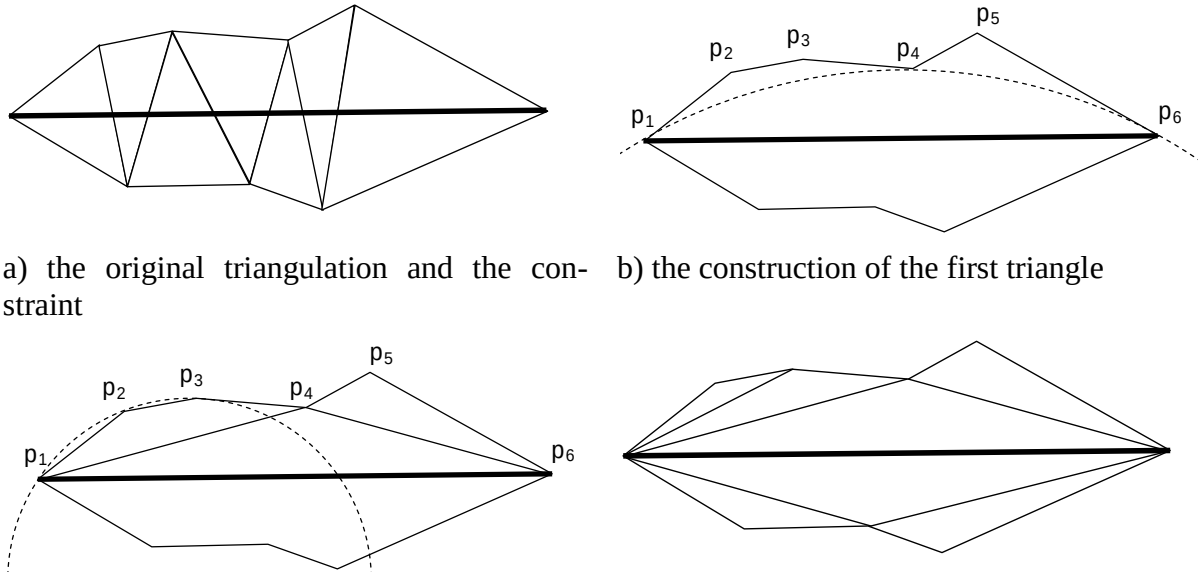
Constrained Delaunay triangulation is a generalization of Delaunay triangulation offering a possibility to incorporate some prescribed edges or faces (i.e., constraints) into the triangulation. Typically, these constraints are used either to express the shape of object whose sampled points are to be triangulated or to introduce some physical limitations. Figure 2.12 compares the Delaunay triangulation and the CDT of the same input set in E^2. The prescribed edges are thick. Constrained Delaunay triangulation is used in many applications, e.g., numerical analysis and finite element methods (FEM), pattern recognition [Pra00, Xia02], etc.



a) DT b) CDT

Figure 2.12: *An example of the Delaunay triangulation (DT) and the Constrained Delaunay triangulation (CDT) in E^2. Prescribed edges are thick.*

From the point of view of the algorithm based on the incremental insertion with local transformations, a constraint is an edge or a face from the triangulation that cannot be flipped, i.e., this edge is always considered valid in the meaning of the Delaunay criterion. This means that the legalization stops on constraints. The best-known approach for the insertion of a constraint into the triangulation works as follows. First, all simplices crossed by this constraint are detected – see Figure 2.13. These simplices are removed from the triangulation, which results in two adjacent holes separated just by the constraint. Then, both holes have to be retriangulated. For this purpose, the ear cutting algorithm that was presented in the section describing the deletion of points from the triangulation can be used.



a) the original triangulation and the constraint b) the construction of the first triangle

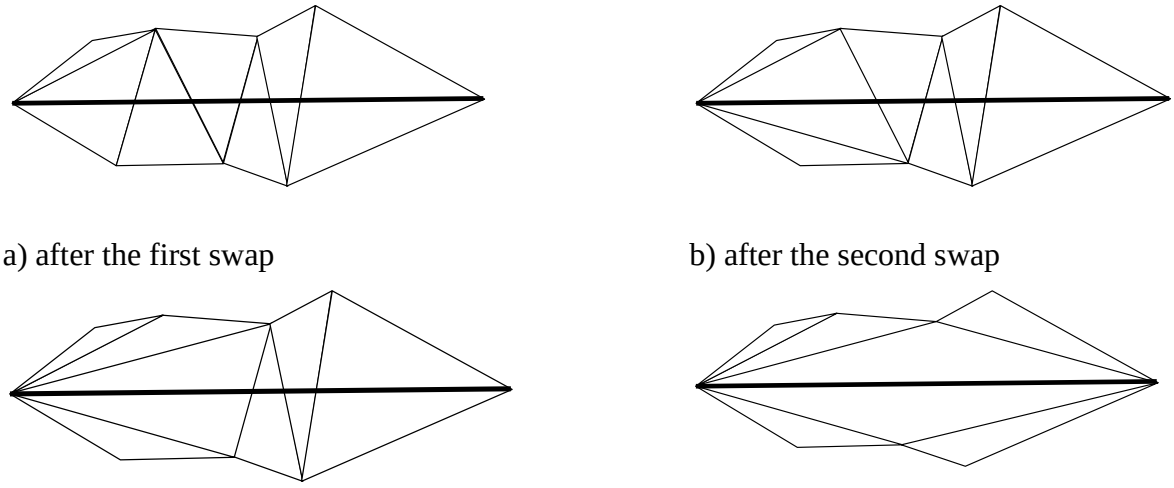
c) the construction of the second triangle d) the result

Figure 2.13: *An example of the insertion of a constraint (thick edge) into Delaunay triangulation in E^2.*

Unlike the deletion of a vertex, all vertices of hole lie in the same half-plane (or half-space) defined by the constraint, which allows us to consider another, much easier, algorithm. It is based on the D&C strategy. Starting with the constraint edge (or face), in each step of the recursion, the algorithm constructs a simplex such that no vertex from the hole (naturally, ex-

cept for the vertices forming the simplex) lies inside the circum-sphere of this simplex. The constructed simplex issues new two edges (faces) and may split the hole into two smaller holes that are retriangulated in next step – see Figure 2.13. At the end of the insertion of the constraint, the connectivity between simplices is updated.

Sloan [Slo92] suggested another algorithm for the insertion of a constraint into the triangulation. Starting from any triangle containing the first vertex of the given constraint, the algorithm searches the triangulation until it reaches the triangle containing the second vertex of the constraint. For each triangle visited during the search, the algorithm checks whether there is an edge intersected by the constraint. If the outcome of this test is positive, the edge is flipped. It can be shown that the successive performing of flips ensures that when the second vertex of the constraint is reached, the constraint is included in the triangulation. Afterwards, the legalization has to be performed in order to restore the Delaunay property of the triangulation. An example of such insertion of a constraint can be seen in Figure 2.14.



a) after the first swap b) after the second swap

c) after the third swap d) the result

Figure 2.14: *An example of the insertion of a constraint (thick edge) into Delaunay triangulation in E^2 using the successive application of local transformations.*

Similarly to regular triangulations, constrained Delaunay triangulations can preserve image feature better than Delaunay triangulation. Its construction is, however, more difficult. Let us note that for a given set of points, it should be theoretically possible to assign a weight to every point in such a manner that the regular triangulation of these points is identical to the required CDT [Mau04]. As the amount of constraints is very low (in comparison with the amount of all edges), it is, however, more convenient to store coordinates of vertices and associated data (like in the case of Delaunay triangulation) followed by the constraints.

2.4 Data Dependent Triangulation

The previously described triangulations triangulate the given set of points without taking the associated data (i.e., grey-scale or colour components values) into account. If the data values change rapidly, e.g., on sharp image edges, this shape information may not be well preserved by these triangulations or its preservation requires lot of vertices. Therefore, Dyn et al. introduced data dependent triangulation [Dyn90] that uses a data dependent criterion instead of the circum-circle criterion (or a similar one). Many criteria have been proposed; some of them consider z-coordinates directly, others deal with angles between triangle normals, etc.

Data dependent triangulation is typically constructed by successive application of local transformations on initial triangulation (e.g., the Delaunay one). Each edge is checked whether its cost is lower than the cost of the other diagonal of the quadrilateral formed by two triangles sharing this edge. If the outcome of this test is negative, the edge is replaced by the other diagonal – see Figure 2.15.

Data dependent triangulation can be considered as a more general regular triangulation where weights of points are not constants given explicitly in advance but they are functions whose values dynamically change during the triangulation process. Once the triangulation is completed, however, the resulting triangulation can be processed (and stored) using the same techniques that are available for regular triangulations.

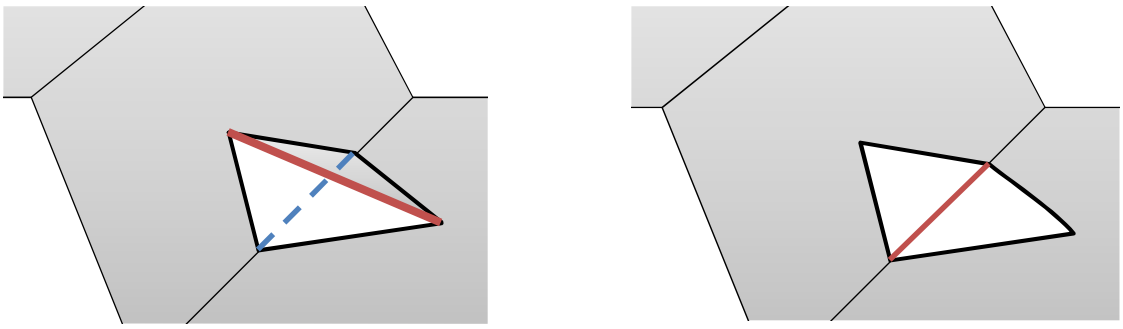
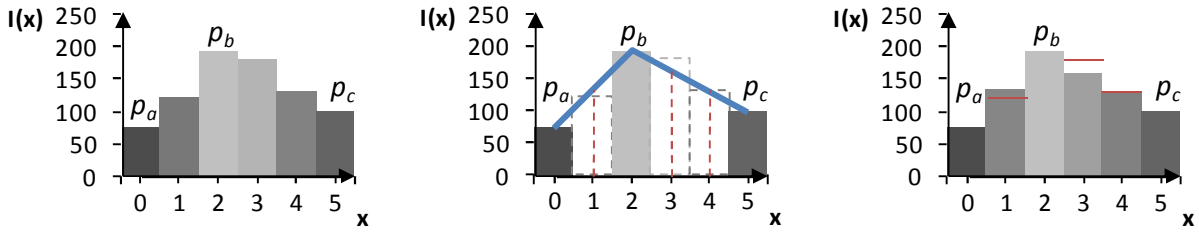


Figure 2.15: *Local transformation of quadrilateral formed by two triangles sharing the bold (red) common edge. Edge is swapped to better preserve the shape of data (slope).*

3 Selection of Significant Points

An image represented by a raster of $N \times M$ pixels can be easily, and without any loss of information, transformed into a point set S in E^2 such that for each pixel there is one point with x and y-coordinates defined by the position of this pixel in raster and with the associated data corresponding to the grey-scale value (or colour components values) of this pixel. Certainly, this transformation is reversible (and there is no loss of information).

Let us suppose that we have a subset S_S of points from S ($S_S \subset S$), and a point set S_M created from the remaining points, i.e., from $S - S_S$, by preserving their x and y coordinates only, which means that the data associated with points from S_M (i.e., grey-scale or colour component values) is unknown. This missing data can be approximated by an interpolation of the associated data of points from S_s (see Figure 3.1), which gives a new set S_R of points. The point set $S' = S_S \cup S_R$ is an approximation of the original point set S.



a) original data b) linear interpolation c) reconstructed data

Figure 3.1: *Linear interpolation of points p_a, p_b and p_c in E^1 and the error of approximation (denoted by horizontal line segments in the reconstructed data).*

It is clear that the approximation error is influenced by the number and distribution of points from the subset S_S and by the interpolation method used for the reconstruction. Interpolation methods can be global, which take all points from S_S into account for the point reconstruction, or local, which consider only those points from S_S that lie near the point to be reconstructed. Although global methods very often achieve lower approximation error than local, for some particular subsets S_S, they produce much worse results than local methods because of their instability. As they consume a lot of time, they are unsuitable for real-time processing. Hence, we decided to use local interpolation methods only.

If the points from the subset S_S are triangulated first, e.g., by the Delaunay triangulation (see the previous section), the detection of points close to the point to be reconstructed can be done in $O(1)$. Certainly, the approximation error depends on the kind of triangulation. If not specified otherwise, from now on, we assume that points are triangulated by the Delaunay triangulation. Using the bilinear interpolation of triangles, the points from S_M can be reconstructed in real-time with an ease. In the further text, we assume, therefore, that the reconstruction is done by this interpolation, if not specified otherwise.

Our task can be defined as follows. We want to find the subset S_S such that either it is the minimal subset from which the point set S' can be reconstructed with an approximation error within the given tolerance ε_T, or it contains the given number of points n and the approximation error of reconstruction from this subsets is lower than the error of reconstruction from any other subset with the same number of points. Let us note that if regular or constrained Delaunay triangulations are to be exploited, weights for these points and constraints must be

also found. The points in the subset S_S are furthermore denoted as the most significant (or also the most important) points.

Even if the Delaunay triangulation, which is unique once points are slightly (within one pixel size) randomly shifted, is to be exploited and, therefore, neither weights nor constrains are required, the problem of finding the optimal subset S_S is NP-hard as it needs to check every subset of S. It is clearly beyond our possibilities to check everything and, therefore, some heuristics is necessary.

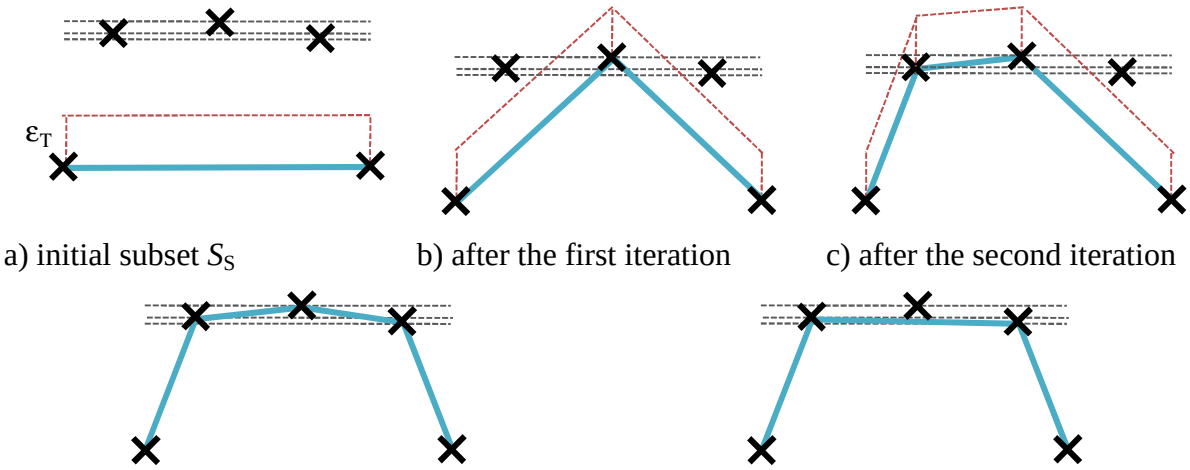
In this section, we describe various heuristics for the detection of the most important points for grey-scale images (the extension for colour images is discussed in Section 7). These methods can be categorized into two main groups. Meshless heuristics compute the significance of a point directly as a function of its grey-scale value and grey-scale values of points in its neighbourhood. All points with the significance larger than some threshold computed from the given tolerance or the first n points (n is given) with the largest significance are taken for the subset S_S. Usually, these heuristics are very fast, however, the produced subset is often too far from being optimal, which means that either this subset is too large or the approximation error is too big. Let us point out that the computation of the threshold is also not well defined.

Starting with an initial subset S_S, mesh based heuristics compute the significance of a point as a function of its influence on the overall approximation error that is achieved for the triangulation of points from the subset S_S. According to the computed significance of points, the subset S_S is modified and the process repeats until the given requirements (i.e., the maximal approximation error or the number of points) is fulfilled. Due to the nature of mesh based heuristics, it is clear that the detection of the most important points and their triangulation are not two independent steps. These heuristics are slower than meshless heuristics (it takes some time to compute the triangulation) but they can achieve better results than their counterparts.

There are four different strategies how to choose the initial subset and modify the current subset when significances of points are recalculated. In the refinement, the initial subset S_s contains only four points representing the corners of the image and this subset is successively modified by adding the most significant point, not already included in this subset. This strategy is useful especially in cases when the compactness of the produced geometrical representation is desired. The decimation works in an opposite way. It starts with the initial subset containing all points from S from which it removes successively the least significant points. This strategy is welcome when the quality of the final geometrical representation is preferred to its compactness.

The problem common to both techniques we have just described is that as the subset S_S, and consequently its triangulation, changes, some of points inserted into / removed from this subset in previous iterations may no longer be significant / insignificant. This is illustrated in Figure 3.2. The third point, which was evaluated to be the most significant one in the first iteration, becomes insignificant after the third iteration as the original data can be approximated with the error within the given tolerance without its consideration. The remedy for this problem is to combine both strategies together. Let us note that a mechanism preventing infinite loop of insertion and deletion of the same point is required.

The last strategy starts with a random initial subset of n points that is modified during the iterative process by a genetic or a simulated annealing approach as follows. Points that are apparently not good candidates for the most significant ones are removed from the current subset S_S and randomly chosen points lying in their vicinity are included into the subset. In the simulated annealing, the vicinity area decreases with every iteration. Let us note that this strategy needs a big number of iterations and its behaviour strongly depends on a large set of parameters such as the temperature for the annealing, etc.

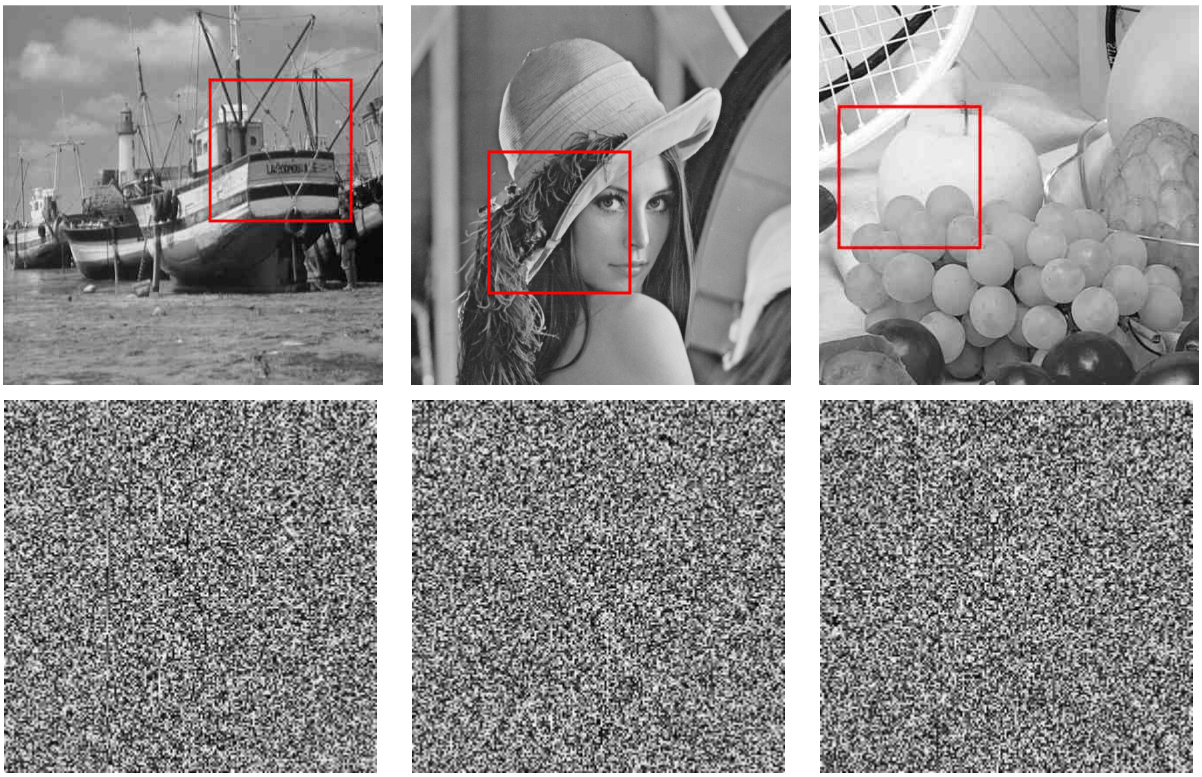


a) initial subset S_S b) after the first iteration c) after the second iteration

d) after the third iteration e) optimal subset S_S

Figure 3.2: *A refinement process in E^1 producing unnecessarily large subset of the most significant points S_S for the given tolerance ε_T of the overall approximation error (depicted by red dashed lines).*

In the further text, we suppose that the described mesh based heuristics exploit the decimation strategy, if not expressed explicitly otherwise.



a) boat, 512x512 *b) Lena, 512x512* *c) fruits, 512x512*

Figure 3.3: *The significance map computed by the RND method for small areas (highlighted by red rectangles) of three popular grey-scale images. Lighter pixels represent more significant points.*

3.1 Random Choice (RND)

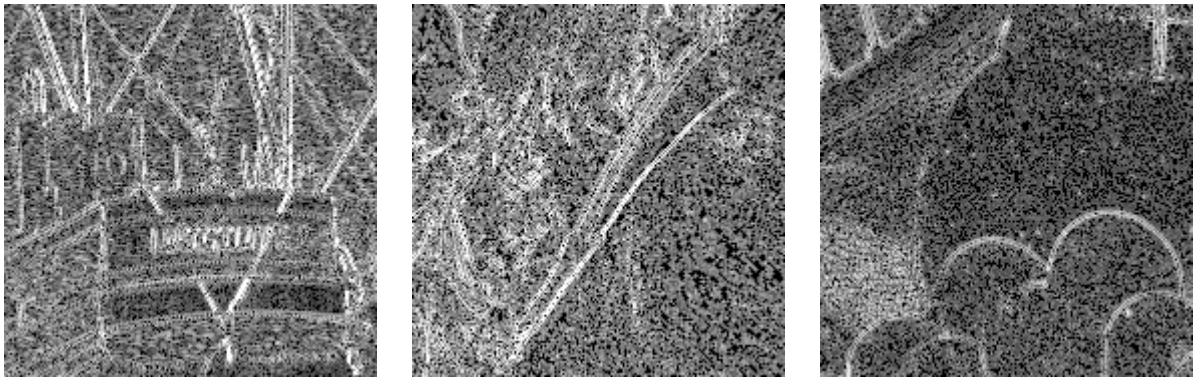
The simplest heuristics, denoted as RND, assigns a random significance to points at the beginning of the decimation process and does not modify it during the process. Actually it means that points are removed from the triangulation in a random fashion. This heuristics can be used in both meshless and mesh based versions and it is not limited to the decimation strategy only. Figure 3.3 shows the significance of points in three popular grey-scale images. As the RND heuristics does not exploit the shape information encoded in images, it is unlikely to achieve good results with this method.

3.2 Marr-Hildreth (MARR)

A more sophisticated method, called MARR, computes the significance of points as the results of Marr-Hildreth edge detection operator [Mar80], which is also known as the Laplacian operator, i.e., points that form edges in the image are more significant. Formally, the significance $s(p)$ of the point p can be defined by the formula:

$$s(p(x,y)) = |I(x-1,y) + I(x+1,y) + I(x,y-1) + I(x,y+1) - 4 \cdot I(x,y)|,$$

where $I(x, y)$ is the grey-scale value of point with coordinates x and y. As in the RND method, the significance is not recalculated during the process and the method is also suitable for any strategy (not only for the decimation). Significances of points for three popular images are shown in Figure 3.4.



a) boat, 512x512 *b) Lena, 512x512* *c) fruits, 512x512*

Figure 3.4: *The significance map computed by the MARR method for small areas of three popular grey-scale images. Lighter pixels represent more significant points.*

3.3 Pixel Similarity (PIXSIM)

The PIXSIM method, another meshless heuristics, is based on the evaluation of similarity between points (pixels). In the following description of this method, we consider that two points are adjacent if and only if they lie on a common horizontal or vertical line and there is no other point on this line that lies between them. In other words, the difference of coordinates of these two adjacent points can be either (1, 0), (-1, 0), (0, 1) or (0, -1). We further define that two points are similar if and only if the absolute value of the difference of their grey scale values does not exceed some given tolerance.

For each point p, the method searches for every point q ($p \neq q$) such that this point and the point p are similar and, moreover, they are either adjacent or they lie on a common horizontal or vertical line and all points on this line lying between them are similar to the point p. In fact,

this corresponds to the region filling algorithm with the given tolerance and the seed in the point *p*. The significance *s* of the point *p* is then calculated as:

$$s(p(x,y)) = \frac{1}{|Q|},$$

where *Q* is a point set including the point *p* and all detected points *q*. Figure 3.5 shows the influence of the tolerance value on significances of points.

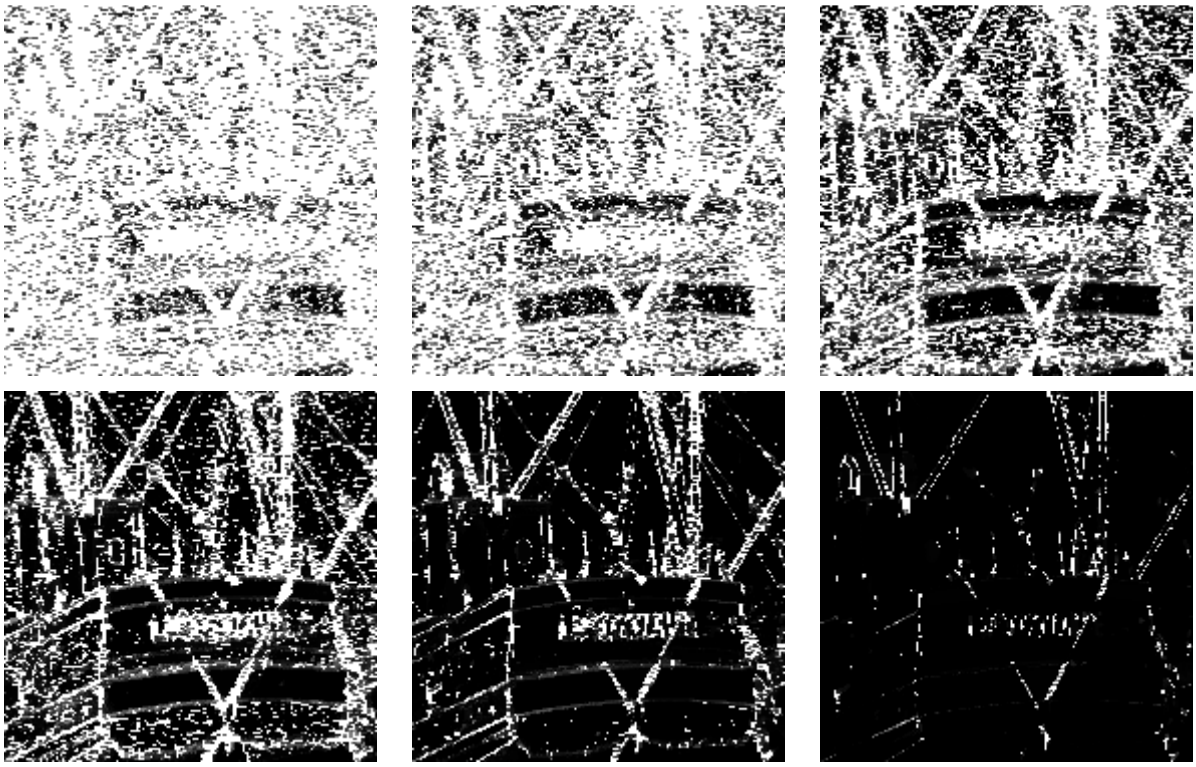
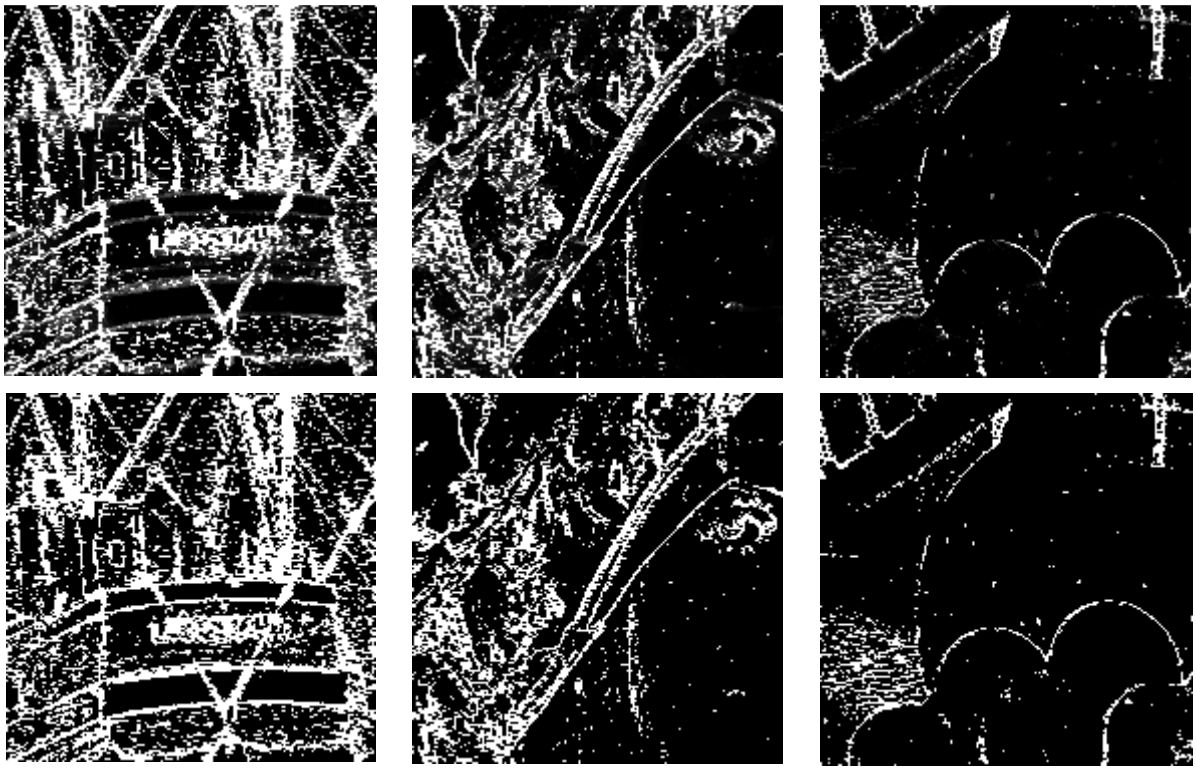


Figure 3.5: *The significance map computed by the PIXSIM method for a small area of the boat grey-scale image when the tolerance 1, 2, 4, 8, 16 and 32 (from top left) was used. Lighter pixels represent more significant points.*

The problem with the previous formula is that it does not consider the position of points, which means that a point surrounded by other eight points with the same grey scale value has the same significance as an endpoint of line segment formed by nine points in total (all points have the same grey scale value). Therefore, we propose also an alternative formula for the evaluation of point significance:

$$s(p(x,y)) = \frac{|Q|}{A(Q)},$$

where *A*(*Q*) is the area of bounding box of the set *Q*. In the further text, we denote this variant by the codename PIXSIM2. A comparison of significances for three popular images is given in Figure 3.6. Let us note that significances of points are not recalculated during the process.



a) boat, 512x512 *b) Lena, 512x512* *c) fruits, 512x512*

Figure 3.6: *The significance map computed by the PIXSIM (top) and PIXSIM2 (bottom) methods using the tolerance value 8 for small areas of three popular grey-scale images. Lighter pixels represent more significant points.*

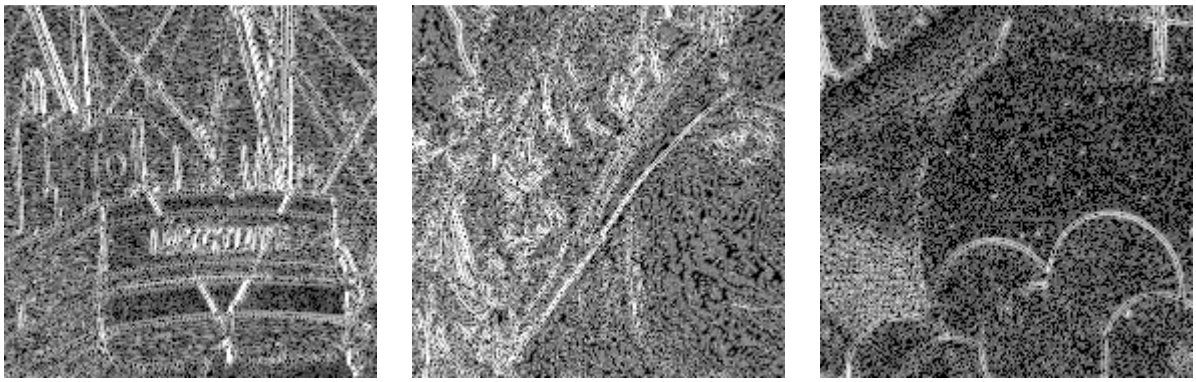
3.4 Distance Weighted (DISTW)

Distance weighted method belongs to mesh based heuristics. It computes the significance of a point *p* as the absolute difference of the grey value of this point and the value computed as the distance weighted average of grey values of its neighbouring vertices *q* that are connected with *p* by an edge:

$$s(p(x,y)) = \left| I(p) - \frac{\sum_q I(q) \cdot |p-q|}{\sum_q |p-q|} \right|.$$

As the point *p*(*x*, *y*) is connected in the initial triangulation, which contains all points from *S*, with points at (*x* - 1, y), (*x* + 1, *y*), (*x*, y – 1), (x, y + 1), (x – 1, y + 1) and (x + 1, y – 1), i.e., four of these points are the same as those used in the MARR method, the initial significance map is very similar to the one obtained by the MARR heuristics – see Figure 3.7.

When a point (the least significant one) is deleted from the triangulation, significances of all points that formed the hole, i.e., they were originally connected by an edge with the deleted point, are recalculated using the formula written above.



a) boat, 512x512 *b) Lena, 512x512* *c) fruits, 512x512*

Figure 3.7: *The significance map computed by the DISTW method for small areas of three popular grey-scale images. Lighter pixels represent more significant points.*

3.5 Error Distribution (ERRDIST)

This heuristics computes the initial significance of points using the same formula as the previously described DISTW method but when the least significant point is deleted from the triangulation, significances of points are updated differently as follows. The triangle containing the removed point p_r is located and significances of its vertices p_a, p_b and p_c – see Figure 3.8 – are modified using the following formulas:

$$s\big(p_a(x,y)\big) = s_{prev}\big(p_a(x,y)\big) + \alpha \cdot s\big(p_r(x,y)\big),$$
$$s\big(p_b(x,y)\big) = s_{prev}\big(p_b(x,y)\big) + \beta \cdot s\big(p_r(x,y)\big),$$
$$s\big(p_c(x,y)\big) = s_{prev}\big(p_c(x,y)\big) + \gamma \cdot s\big(p_r(x,y)\big),$$

where $s_{prev}\big(p_i(x,y)\big)$ is the previous significance associated with the point p_i and α, β and γ are the barycentric coordinates of the point p_r in the triangle p_a, p_b and p_c. These coordinates can be computed from this system of linear equations:

$$\alpha \cdot p_a + \beta \cdot p_b + \gamma \cdot p_c = p_r,$$
$$\alpha + \beta + \gamma = 1.$$

Let us note that this heuristics, actually, works in a way similar to the Floyd-Steinberg error diffusion technique [Flo76].

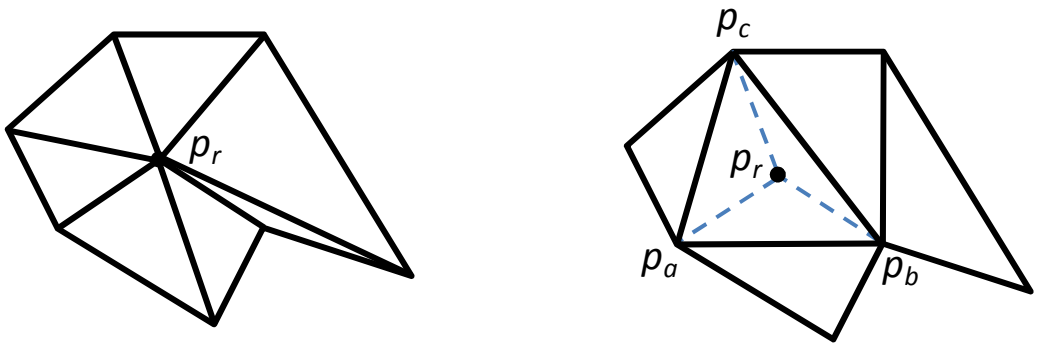


Figure 3.8: *The deletion of point from the triangulation.*

3.6 Triangle Mean Square Error (TRIMSE)

Starting with initial significances of points computed by the formula as the DISTW method, this heuristics updates significances of points, after the least significant point is removed, as follows. For each newly constructed triangle, it first computes its mean square error (MSE), i.e., it computes the sum of square differences between the grey values of pixels covered by this triangle in the original image and the corresponding values obtained by the bilinear interpolation of grey values of triangle vertices that is divided by the number of pixels in this triangle. Formally, this can be written as:

$$MSE_{\Delta} = \sum_{\Delta} \frac{(I(x,y) - I'(x,y))^2}{size(\Delta)},$$

where *I'* denotes the interpolated values. The significance of a vertex is then recalculated as the sum of MSE of triangles that share this vertex. Let us note that this method is, indeed, slower than the previous methods.

3.7 Brute-force (BRUTE)

The BRUTE method is based on a brute-force idea to calculate the significance of a point *p* as a function of the approximation error achieved for triangles that would be constructed if this vertex was deleted. It is clear that there are many possible functions. Whilst the basic function sums the MSE of these triangles (see the previous section), the function denoted as MAT, originally proposed by Demaret et al. [Dem06], sums the square differences between the grey values of pixels covered by these triangles in the original image and the corresponding reconstructed values. This actually means that the MAT computes the sum of MSE of triangles multiplied by the number of pixels they cover. The other two functions, we have experimented with, denoted as MAT1 and MAT2 (both proposed also by Demaret et al.), return the absolute maximal difference and the difference at the point *p*, respectively. In the further text, we assume that the basic function was used unless specified otherwise.

a) boat, 512x512 *b) Lena, 512x512* *c) fruits, 512x512*

Figure 3.9: *The significance map computed by the BRUTE method for small areas of three popular grey-scale images. Lighter pixels represent more significant points.*

It is without any doubt that this method is the most time consuming but we may expect the best results. To speed up the processing, the initial significance of a vertex p is computed simply as the square difference of the grey value held by this vertex and the average of grey values held by two its neighbouring vertices (left and right):

$$s(p(x,y)) = \left(I(p) - \frac{I(x-1,y)+I(x+1,y)}{2}\right)^2.$$

Let us note that this corresponds to the square approximation error measured in this vertex. Initial significances for three popular images are shown in Figure 3.9.

3.8 Nock Segmentation (NOCK)

This heuristics is an extension of the BRUTE method. It computes the initial significance of points in a slightly different way as follows. First, the input raster image is segmented using the technique proposed by Nock et al. [Noc05], which produces several irregular regions of similar pixels. Every point on the boundary of these regions is marked to be not removable, i.e., its significance is set to be maximal. Let us note that the boundary may be thicker than one pixel. An example of the segmentation is given in Figure 3.10. Significances of the remaining points are calculated as in the BRUTE method. The aim of this significance evaluation is to preserve such edges that define the shape encoded in the image.

a) boat, 512x512 *b) Lena, 512x512* *c) fruits, 512x512*

Figure 3.10: *The segmentation of three popular grey-scale images using the method proposed by Nock et al. [Noc05]. The boundary pixels are white.*

3.9 Gaussian Influence (GAUSS)

Another extension of the BRUTE method computes the initial significance of a point p as the sum of Gaussian weighted differences between its grey-scale value and the values of points in its vicinity:

$$s\big(p(x,y)\big) = \sum_{i=-r}^{r}\sum_{j=-r}^{r} |I(x,y) - I(x+i,y+j)| \cdot e^{-\frac{i^2+j^2}{2\cdot\sigma^2}},$$

where constants r and σ define the vicinity area and the influence factor of the point in this area, respectively. This formula is based on the idea that when points in the vicinity of the point p were removed and must be reconstructed, it is likely that points closer to the point p will be influenced by its value more than distant points (which might lie even in a triangle not having the point p as one of its vertices).

Considering the formula given above, it can be deducted that points in uniform regions have very low significances while points lying in proximity of image edges are significant ones and their significances depend on their distances to the nearest edge. Indeed, these values are also

dependent on constants r and σ – see Figure 3.11 and Figure 3.12. Initial significances for three popular grey-scale images are given in Figure 3.13. As it can be seen, for smaller constants, significances are quite similar to those obtained by the Marr-Hildreth method but the map is not so noisy, i.e., points belonging to small unimportant image edges are not evaluated to be significant (unlike in the MARR heuristics).

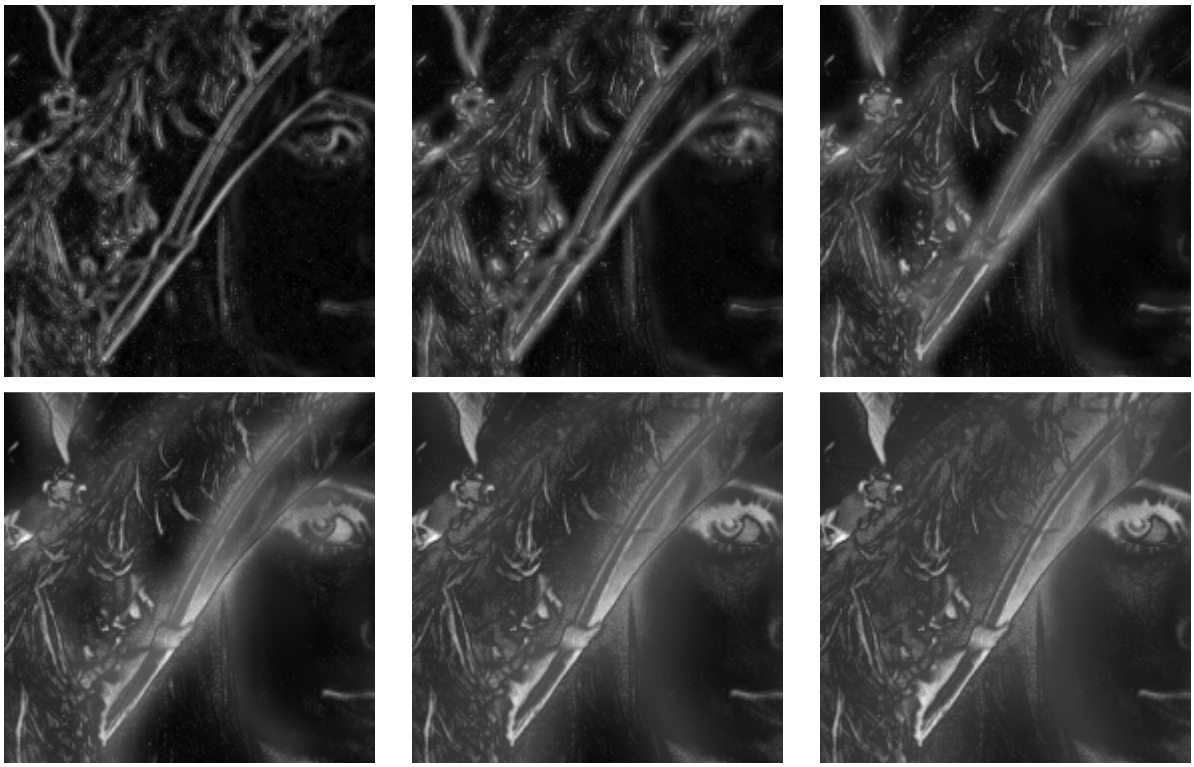


Figure 3.11: *The significance map computed by the GAUSS method for a small area of the Lena grey-scale image when the influence factor (σ) 24 and the vicinity area (r) 2, 4, 8, 16, 32 and 64 (from top left) was used. Lighter pixels represent more significant points.*

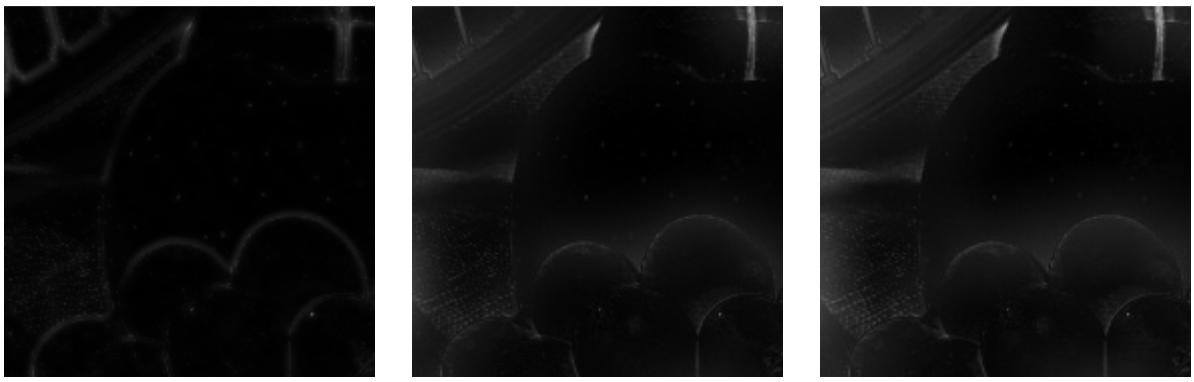
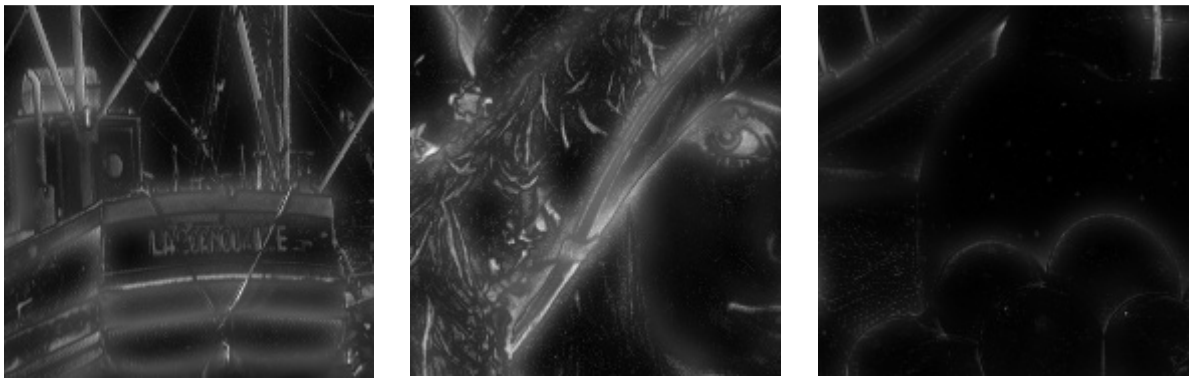


Figure 3.12: *The significance map computed by the GAUSS method for a small area of the fruit grey-scale image when the vicinity area (r) 32 and the influence factor (σ) 2, 16 and 32 (from top left) was used. Lighter pixels represent more significant points.*



a) boat, 512x512 *b) Lena, 512x512* *c) fruits, 512x512*

Figure 3.13: *The significance map computed by the GAUSS method for small areas of three popular grey-scale images when the vicinity area (r) 16 and the influence factor (**σ**) 8 was used. Lighter pixels represent more significant points.*

3.10 Other Heuristics

In this section, we have described various heuristics for the evaluation of point significances. Some of them are complex and tend to be slow (e.g., the GAUSS), however, they promise a more compact representation of images within the given quality threshold. Others are much simpler (e.g. the DISTW method) and offer fast processing, however, they are likely to produce larger representations. Let us note that heuristics described here can be combined to achieve event better results. For instance, one can use the Marr-Hildreth heuristics to evaluate the initial significance of points and then proceed with the BRUTE method, etc.

4 Experiments with Triangulation Construction

For our experiments, we used a set of both grey-scale and colour test images of sizes ranging from 300x400 to 1024x1024 (most of them were of 512x512 size). They were downloaded from the Internet from various sources, e.g., from the USC-SIPI image database [Usc07]. Let us note that most of these images have never been subjected to any lossy compression technique (used, e.g., in JPEG) since their capture into a lossless PNG or TIFF format. Most often tested images from this set are shown in Figure 4.1.

d) boat, 512x512

e) Lena, 512x512

f) fruits, 512x512

g) peppers, 512x512

h) Elaine, 512x512

i) Lukas, 512x512

j) Masa, 512x512

k) Pirate, 1024x1024

l) kodim, 465x375

m) Maran, 400x300

g) monarch, 768x512

Figure 4.1: *Test images.*

For each of proposed methods of the vertex significance evaluation (see the previous section), we investigated the degradation of the quality of the geometric representation in the dependency on the amount of triangulated vertices. The quality was measured as the PSNR, which is computed for images $N \times M$ using the following formula:

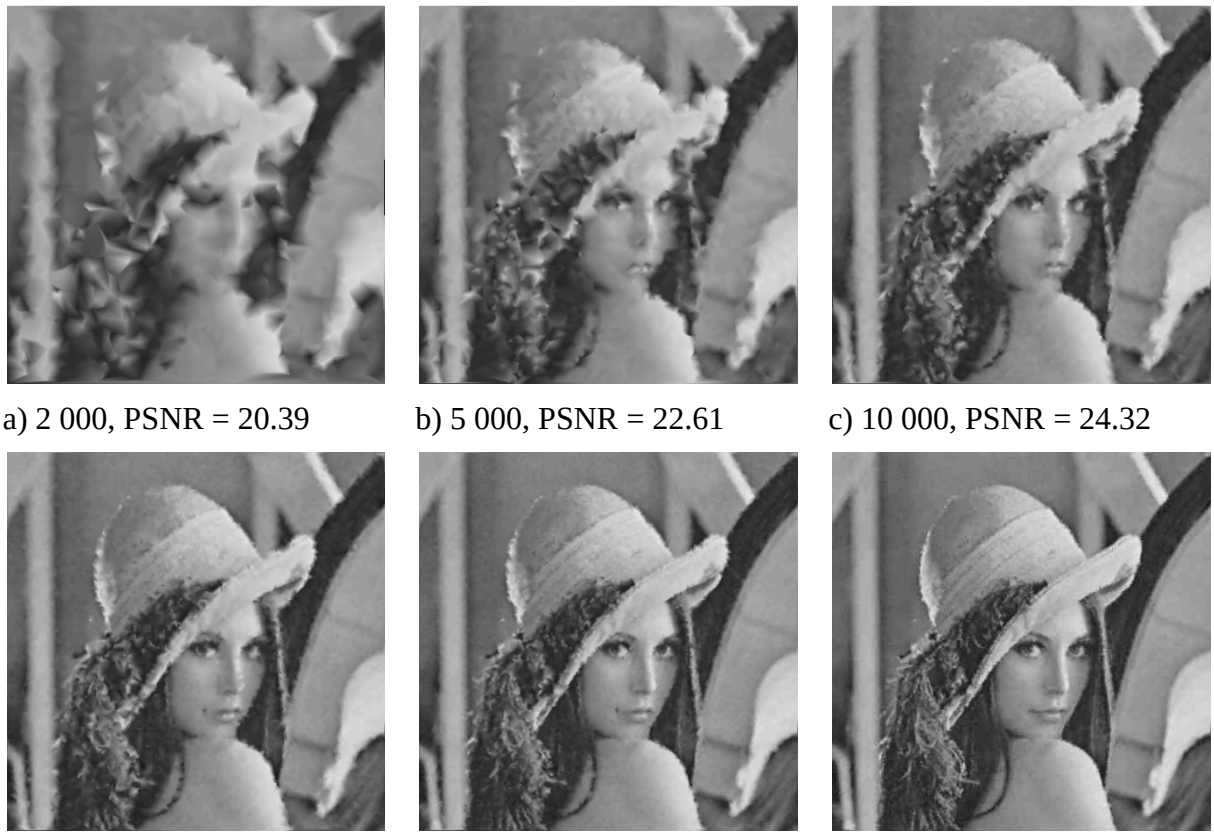
—),

— ,

where $I(x, y)$ is the grey scale value in the original raster image at position x, y while $I'(x, y)$ is the corresponding value in the image reconstructed from the given geometric representation.

4.1 Main Meshless Heuristics

Figure 4.2 brings examples of Lena images reconstructed by the bilinear interpolation on the Delaunay triangulation of points selected by the random choice (the RND method). As it can be seen, edges in images are not smooth and triangles of the underlying triangulation are clearly visible even for triangulations with lot of vertices. On the other hand, even images reconstructed from the Delaunay triangulation of a very few vertices (e.g., 1.9% as in Figure 4.2b) are well recognisable. Apparently, the RND heuristics could be used for very fast previews or for non-realistic rendering (especially, for the pointillism).

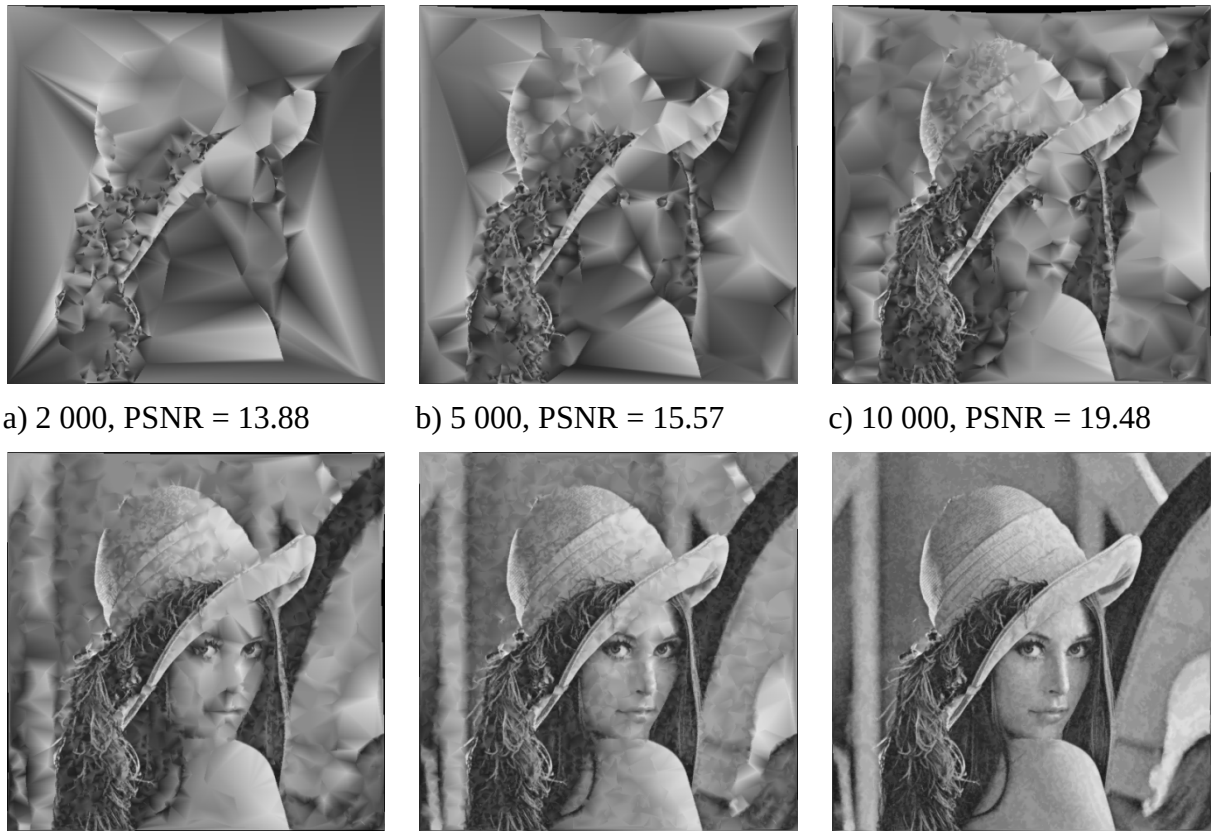


a) 2 000, PSNR = 20.39 b) 5 000, PSNR = 22.61 c) 10 000, PSNR = 24.32

d) 15 000, PSNR = 25.30 e) 25 000, PSNR = 26.64 f) 50 000, PSNR = 28.86

Figure 4.2: *The comparison of Lena images reconstructed from triangulations with various vertices computed by the RND method.*

The results of the Marr-Hildreth (MARR) method are given in Figure 4.3. Unexpectedly, this heuristics provide us with the results of a low quality (especially for small triangulations). The reason is that there is not a sufficient amount of vertices to represent areas with a smooth change of intensity (e.g., in face) in a good quality because too many vertices were wasted to represent areas with sharp edges in an outstanding quality (see the feather).



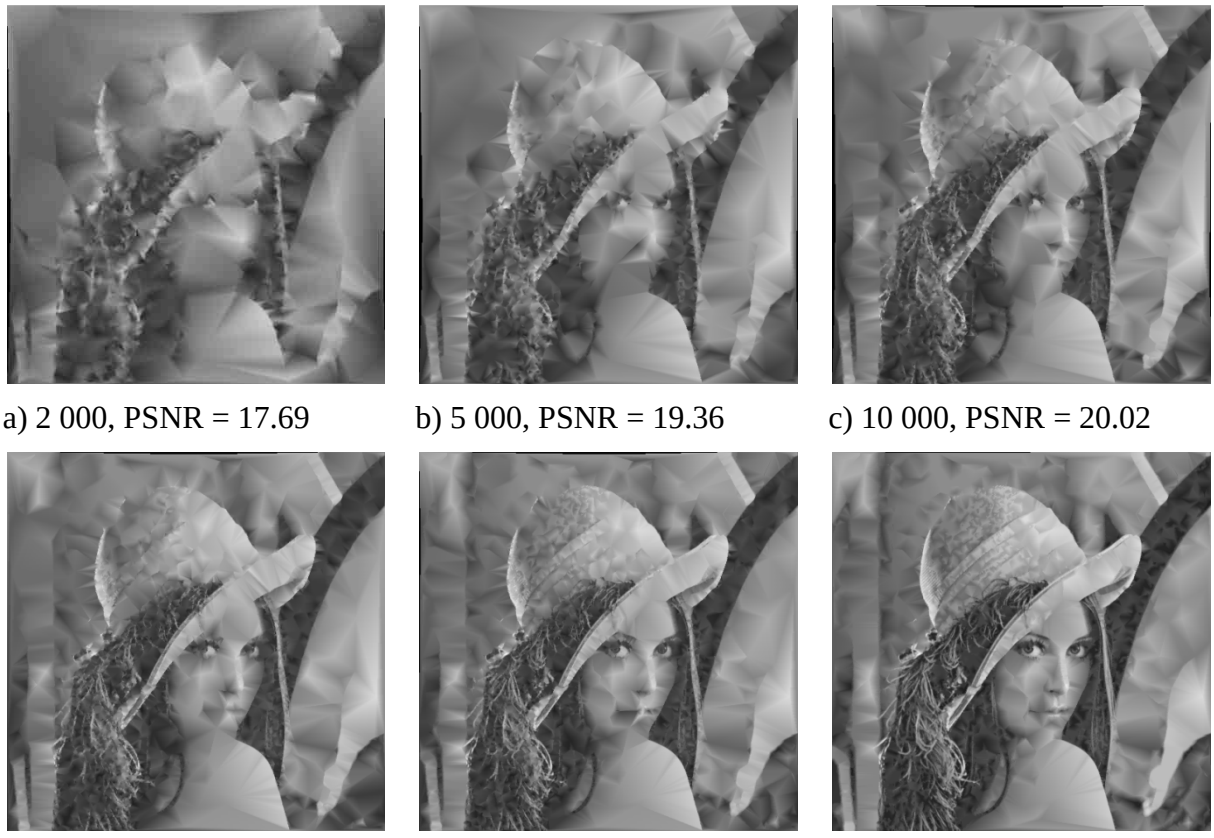
a) 2 000, PSNR = 13.88 b) 5 000, PSNR = 15.57 c) 10 000, PSNR = 19.48

d) 15 000, PSNR = 20.46 e) 25 000, PSNR = 24.93 f) 50 000, PSNR = 30.95

Figure 4.3: *The comparison of Lena images reconstructed from triangulations with various vertices computed by the MARR method.*

There is no significant difference between the PIXSIM and PIXSIM2 heuristics – compare Figure 4.4 and Figure 4.5. It is not a big surprise since significance maps for the tested image are also very similar (see Figure 3.6). However, the issue is whether this can be expected in a general case. Let us analyse both heuristics for different inputs. If we have an image of uniform colour, every point has the same number of adjacent similar points and, therefore, all points are equally significant in both methods. For an artificial image having two regions of the same size but different, not similar, colours (e.g., the left region is white whilst the right region is black), significances of points are also equal. This actually means that these heuristics are impropriate for images with regular patterns not separated by an edge. Fortunately, such patterns are not very often present in real images.

In real images, regions, either gradient or uniform, are always separated by edges at least one pixel thick that typically blends colours of one region into colours of the other region. Let us consider an image with two uniform regions separated by such an edge. For the sake of simplicity, we assume that this image has only one line of pixels (the extension for two-dimensional case is straightforward). It can be considered to be a one-dimensional function that maps x-coordinates into grey-scale values. This function is depicted in Figure 4.6a.



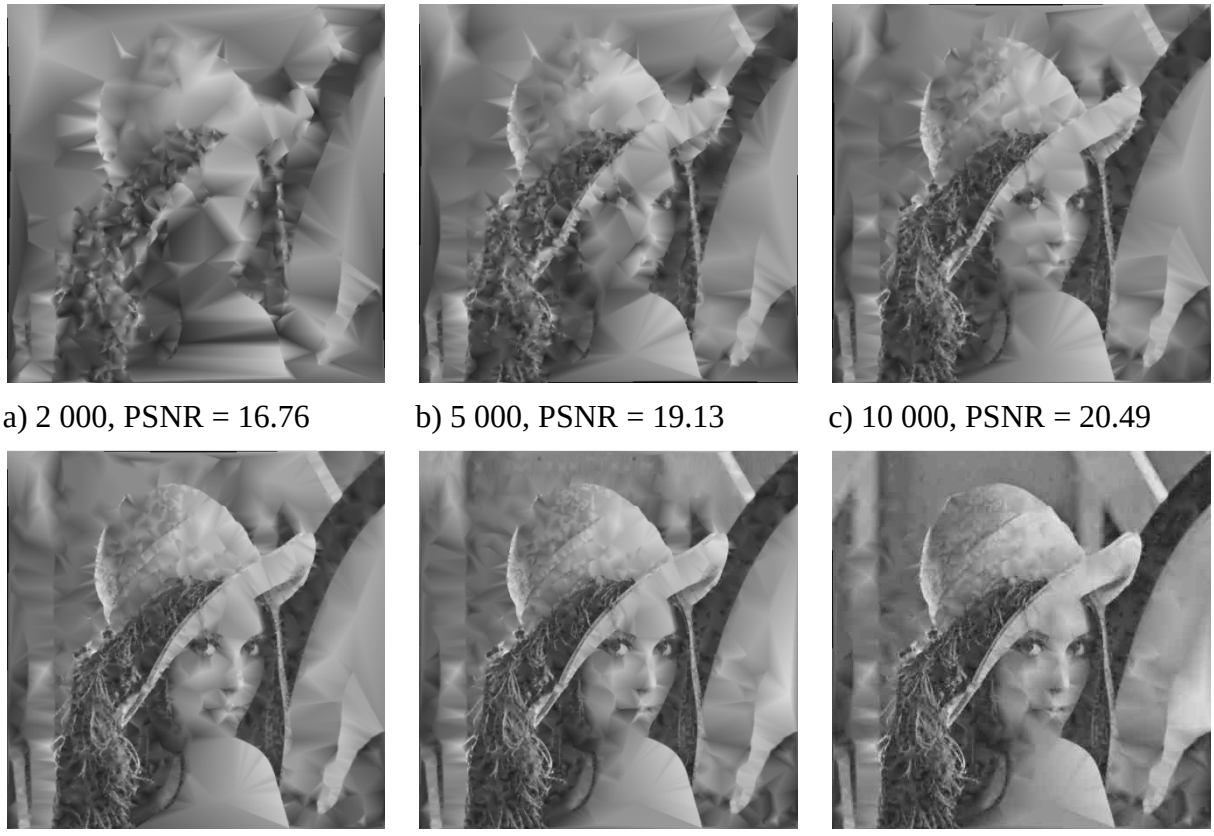
a) 2 000, PSNR = 17.69 b) 5 000, PSNR = 19.36 c) 10 000, PSNR = 20.02

d) 15 000, PSNR = 21.56 e) 25 000, PSNR = 22.19 f) 50 000, PSNR = 24.16

Figure 4.4: *The comparison of Lena images reconstructed from triangulations with various vertices computed by the PIXSIM method with the tolerance value 8.*

If we compute differences between adjacent pixels (i.e., the differential: $I(x + 1) – I(x)$), we get zeros for uniform areas and non-zeroes for the edge with the peak in the middle of the edge – see Figure 4.6b. The algorithm for the searching of similar points adjacent to a point p, which was described in the previous section, can be converted now into another one. This algorithm starts at the point $p(x)$ with the total energy equalled to the tolerance value and it advances to points with lower x-coordinates decreasing this energy in every step by the difference value at the current position. Naturally, the process terminates when there is not enough energy to move to the adjacent point. After that the algorithm is repeated, this time it advances to points with higher x-coordinates.

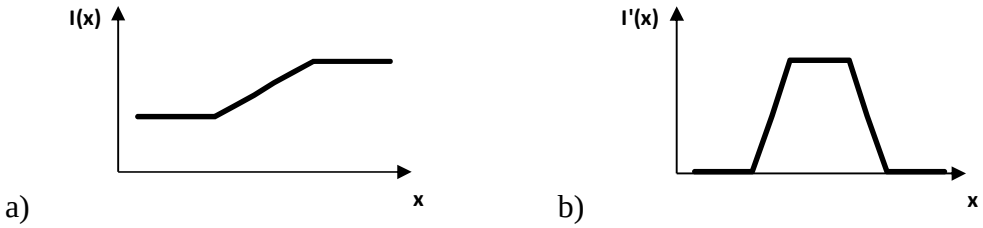
For instance, let us assume that we have a line of pixels with grey-scale values: 1, 2, 3, 4 and 6. The differences are then: 1, 1, 1, 2 and 0 for the last pixel as it does not have the right neighbour. For the tolerance value 2, the region of adjacent pixels similar to the third pixel clearly includes four pixels (1, 2, 3 and 4). The algorithm starts at the third pixel with the total energy 2. As the cost of movement to the second pixel is just one, it moves there and decreases the total energy by one. As it has still enough energy, the algorithm moves then to the first pixel. After that no energy remains and the process is restarted for the right side, where it can move only to the fourth pixel as it lacks energy to move further. Hence, the resulting region contains the same pixels as the region produced by the original algorithm used in the PIXSIM and PIXSIM2 heuristics.



a) 2 000, PSNR = 16.76 b) 5 000, PSNR = 19.13 c) 10 000, PSNR = 20.49

d) 15 000, PSNR = 21.22 e) 25 000, PSNR = 22.91 f) 50 000, PSNR = 24.24

Figure 4.5: *The comparison of Lena images reconstructed from triangulations with various vertices computed by the PIXSIM2 method with the tolerance value 8.*



a) b)

Figure 4.6: *An example of one-dimensional function I(x) and its derivation I'(x).*

Being at the point $p(x)$ with an unknown amount of energy, the probability that the algorithm can move to the adjacent point is inversely proportional to the difference between grey-scale values of these points. It actually means that the amount of similar points adjacent to the given one is highly dependent on the shape of the function of differences. If the slope around the point is small, the amount is much larger than in the case of big slopes. As image edges produce sharp peaks (and, therefore, big slopes) – see also Figure 4.7, we can conclude that the number of similar adjacent points is small for an image edge while it is large for other areas. Hence, the most significant points in the PIXSIM heuristics are those that represent

image edges. Further, it can be easily shown that the ratio between the amount of similar points adjacent to the given one and the area of bounding box of these similar points typically decreases as this amount grows. It means that this ratio is larger for points of image edges, which leads into conclusion that the PIXSIM2 method also assigns higher significances for points that represent image edges. Hence, both heuristics produce almost the same results for any digital image.

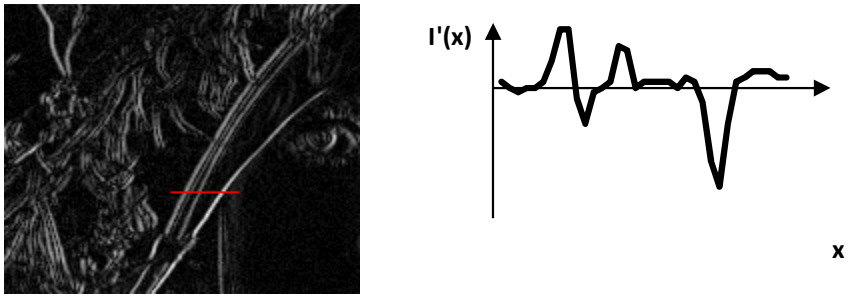
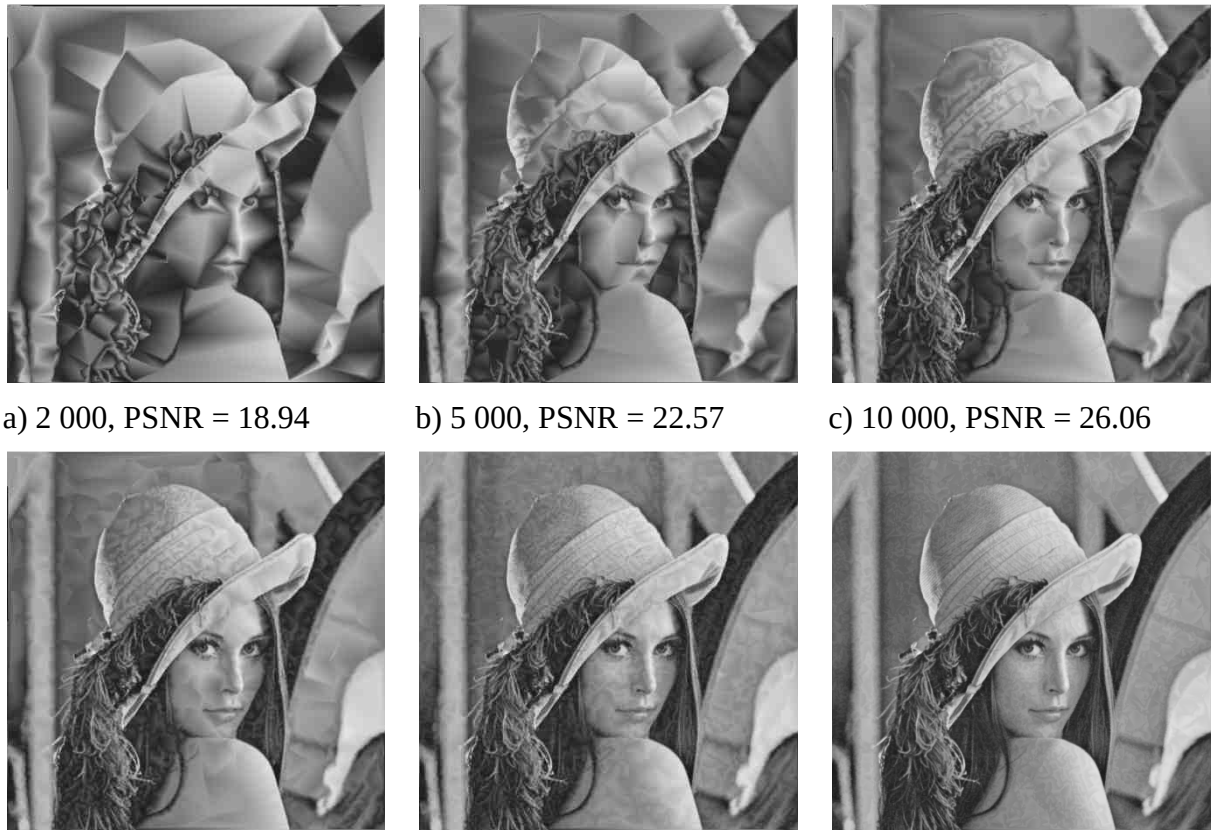


Figure 4.7: *Differences in x-coordinate for a small part of the Lena image. Red line of pixels is given in detail in the right image.*

From the presented results, it is apparent that the PIXSIM and PIXSIM2 heuristics can preserve the image shape better than the MARR method in the case of triangulations of a very few vertices. However, they still cannot beat the simplest method of random choice.



a) 2 000, PSNR = 18.94 b) 5 000, PSNR = 22.57 c) 10 000, PSNR = 26.06

d) 15 000, PSNR = 28.22 e) 25 000, PSNR = 30.86 f) 50 000, PSNR = 33.38

Figure 4.8: *The comparison of Lena images reconstructed from triangulations with various vertices computed by the DISTW method.*

4.2 Main Mesh Based Heuristics

Figure 4.8 shows the results for the first mesh based heuristics, the DISTW. This method apparently inclines to preserve those points that represent image edges. Unlike the meshless MARR heuristics, the weighting scheme prevents the method from the keeping of improper points and, therefore, the quality of achieved results is better (much better for smaller triangulations). The most important information is preserved even if the image is represented by a tiny triangulation (see Figure 4.8a). Nevertheless, the quality is still far from being acceptable.

Similar results were achieved for the ERRDIST heuristics – see Figure 4.9. As it can be seen, this heuristics also tends to keep points on image edges. In comparison to the previous method, the significance of those points is, however, a bit diminished due to the error distribution scheme, which may lead to slightly better results for smaller triangulations. Let us note that both methods are very fast (although they are slower than meshless methods), yet they can represent images in an acceptable quality retaining only about 6% of their pixels. Supposing that no compression technique is involved, the storage cost for this geometrical representation is, therefore, about one third of the storage cost for the raster representation.

a) 2 000, PSNR = 19.82 b) 5 000, PSNR = 20.35 c) 10 000, PSNR = 22.53

d) 15 000, PSNR = 26.72 e) 25 000, PSNR = 27.57 f) 50 000, PSNR = 32.08

Figure 4.9: *The comparison of Lena images reconstructed from triangulations with various vertices computed by the ERRDIST method.*

The results achieved by the TRIMSE heuristics are given in Figure 4.10. As this method evaluates the significance of point according to the mean square error of triangles sharing this point, there is a little wonder that reconstructed images, especially those reconstructed from smaller triangulations, have visible triangular artefacts. Unlike the RND method, which also produces similar artefacts, the results are, however, of much better quality. It is also important

to point out that this heuristics can achieve better quality for moderate triangulations than the previously described methods. Nevertheless, the images reconstructed from such triangulations are a bit blurred (and with visible triangular artefacts, indeed).

a) 2 000, PSNR = 21.16 b) 5 000, PSNR = 24.58 c) 10 000, PSNR = 27.09

d) 15 000, PSNR = 28.34 e) 25 000, PSNR = 29.69 f) 50 000, PSNR = 31.25

Figure 4.10: *The comparison of Lena images reconstructed from triangulations with various vertices computed by the TRIMSE method.*

Figure 4.11 and Figure 4.12 show the results achieved by all considered variants of the BRUTE heuristics. Let us remind the reader that the MAT, MAT1 and MAT2 versions were originally proposed by Demaret et. al [Dem06]. Except for the MAT2 variant, which is the fastest one, the results are of an outstanding quality. This is definitely true for the basic and the MAT versions that can represent images in a good quality retaining only about 5 000 pixels (2%), which reduces the storage cost to one tenth in comparison with the raster representation (assuming that no compression technique is involved).

As it can be seen, there is no significant difference between the basic and the MAT variants of the BRUTE heuristics. This is hardly surprising, since both variants compute significances of points in a very similar way. An interesting observation, however, is that the results obtained by the MAT2 version are not significantly better (in the meaning of achieved quality) than results obtained by much faster distance weighted heuristics (DISTW).

A serious drawback of the BRUTE heuristics is its time consumption. Even the fastest version takes almost one minute to process an image of 512×512 pixels and, therefore, this method cannot be used in real-time (or almost real-time) applications. However, we believe that with a careful implementation exploiting modern hardware (such as GPU for interpolation of triangles), it could be possible to reduce the time requirements to mere seconds.

a) 2 000, PSNR = 24.52 b) 5 000, PSNR = 28.57 c) 10 000, PSNR = 31.13

d) 15 000, PSNR = 32.53 e) 25 000, PSNR = 34.31 f) 50 000, PSNR = 36.30

g) 2 000, PSNR = 24.98 h) 5 000, PSNR = 29.83 i) 10 000, PSNR = 31.72

j) 15 000, PSNR = 32.74 k) 25 000, PSNR = 33.94 l) 50 000, PSNR = 35.74

Figure 4.11: *The comparison of Lena images reconstructed from triangulations with various vertices computed by the basic version of the BRUTE method (a-f) and the MAT version (g-l).*

a) 2 000, PSNR = 23.97 b) 5 000, PSNR = 27.94 c) 10 000, PSNR = 30.68

d) 15 000, PSNR = 32.11 e) 25 000, PSNR = 33.53 f) 50 000, PSNR = 35.45

a) 2 000, PSNR = 22.01 b) 5 000, PSNR = 25.47 c) 10 000, PSNR = 28.37

d) 15 000, PSNR = 28.37 e) 25 000, PSNR = 29.87 f) 50 000, PSNR = 33.49

Figure 4.12: *The comparison of Lena images reconstructed from triangulations with various vertices computed by the MAT1 version of the BRUTE method (a-f) and the MAT2 version (g-l).*

The results achieved by the NOCK method (based on the BRUTE) are given in Figure 4.13. As it can be seen, the quality of reconstructed images quickly degrades with the decreasing size of the triangulation. The reason of this behaviour is that too many points represent region edges and, therefore, there is hardly any other point left to preserve the data within regions. This is well visible especially in Figure 4.13a. This problem might not be so severe for the Lena image that has only about 7 000 boundary pixels but it is a serious issue for complex images with plenty of details. A possible solution could be to decimate boundaries first, i.e., to reduce the number of boundary pixels.

a) 7 017, PSNR = 16.43 b) 10 017, PSNR = 27.42 c) 15 017, PSNR = 30.93

a) 19 517, PSNR = 32.37 b) 27 017, PSNR = 33.94 c) 47 017, PSNR = 35.98

Figure 4.13: *The comparison of Lena images reconstructed from triangulations with various vertices computed by the NOCK method (with one pixel thick boundaries).*

The GAUSS heuristics, which is also based on the BRUTE heuristics, unfortunately does not bring any further improvement – see Figure 4.14. The results are rather worse than better. Considering that this heuristics needs $O(N \cdot M \cdot 4r^2)$ time for the computation of initial significances, where N and M are numbers of rows and columns in the image and r is the vicinity area, which is typically 16, it is a big disappointment. We tried, therefore, to combine this heuristics with the ERRDIST one in such a manner that initial significances of points are computed using the algorithm of the GAUSS method and then the error distribution decimation algorithm is applied. For smaller triangulations, the achieved results are of a better quality than those obtained by pure ERRDIST method – compare Figure 4.15 and Figure 4.9. Hence, this combined method is apparently suitable for larger images whose processing using BRUTE or TRIMSE methods would take too long but using much faster DISTW and ERRDIST methods would not meet the requested quality criterion.

a) 2 000, PSNR = 22.13 b) 5 000, PSNR = 28.35 c) 10 000, PSNR = 30.99

d) 15 000, PSNR = 32.34 e) 25 000, PSNR = 33.85 f) 50 000, PSNR = 35.42

Figure 4.14: *The comparison of Lena images reconstructed from triangulations with various vertices computed by the GAUSS method with when the influence factor (**σ**) 8 and the vicinity area (r) 16.*

a) 2 000, PSNR = 22.43 b) 5 000, PSNR = 25.09 c) 10 000, PSNR = 27.07

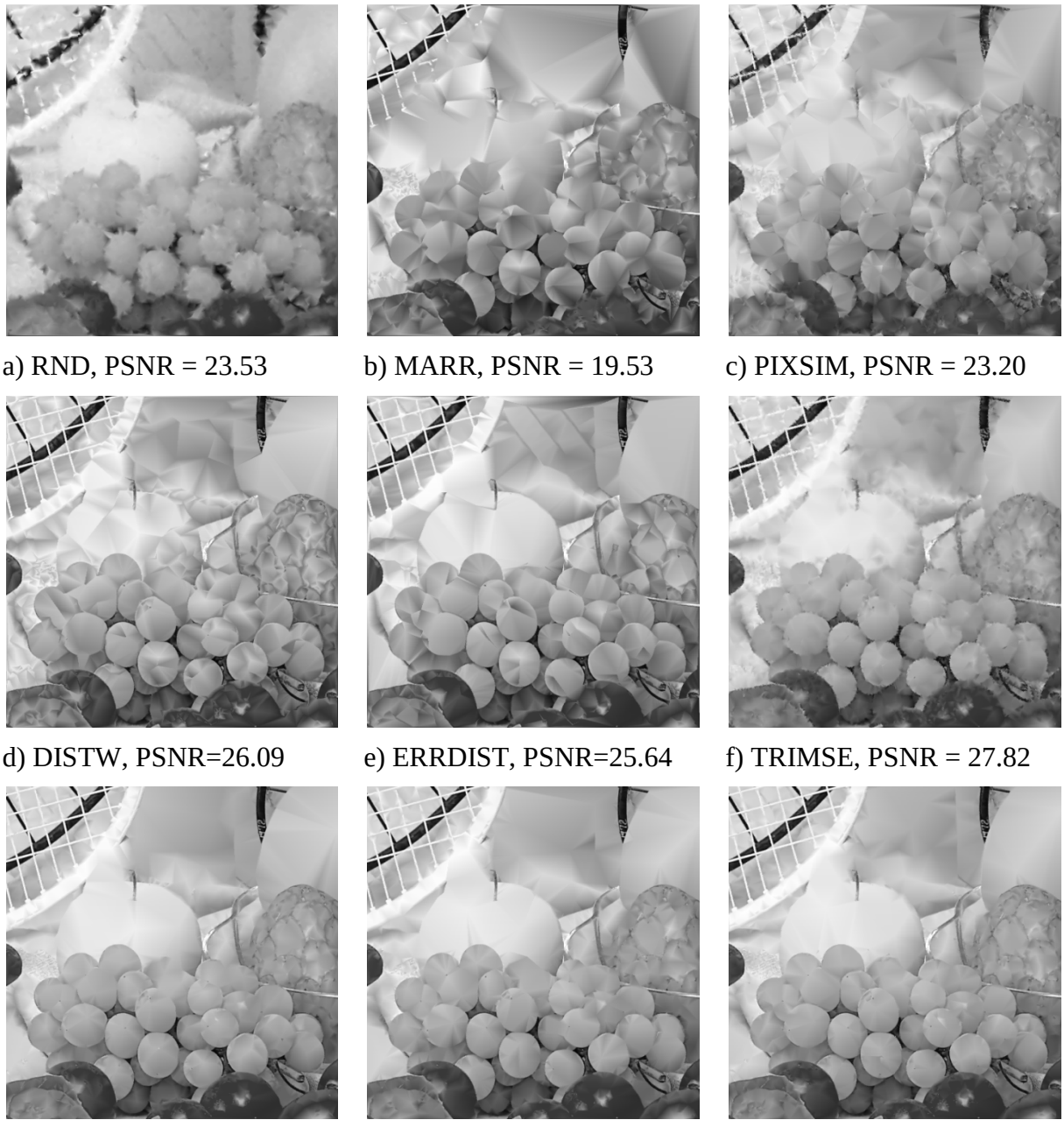
d) 15 000, PSNR = 28.39 e) 25 000, PSNR = 29.88 f) 50 000, PSNR = 31.71

Figure 4.15: *The comparison of Lena images reconstructed from triangulations with various vertices computed by the ERRDIST method combined with the GAUSS method with when the influence factor (**σ**) 8 and the vicinity area (r) 16.*

4.3 Comparison of Main Heuristics

Figure 4.16 compares images of fruits reconstructed from Delaunay triangulations with 10 000 vertices (i.e., 96% of vertices was removed) computed using different methods for the evaluation of vertex significance. Without any doubt, the best results were obtained by the BRUTE method. However, it is interesting to point out that this method did not preserve some details that might be important from the user point of view. For instance, see missing diagonal lines on the background. On the contrary, the ERRDIST method, although not so powerful as the BRUTE one, preserved it quite well. Hence, a smart combination of these two methods seems to be optimal.

For each proposed heuristics, we investigated the degradation of the quality of the geometric representation in the dependency on the amount of removed points (the number of significant vertices in the Delaunay triangulation can be calculated as the total number of points in the original image minus this value). The results for three popular images, all with 262 144 points in total, are presented in Figure 4.17 – Figure 4.20. As it can be seen from graphs, the quality of the representation degrades quite quickly until the algorithm removes approximately 25% of vertices, after that the quality decreases almost linearly in a slow pace until another threshold of about 90% removed vertices is reached. From that moment, the quality rapidly drops down. An interesting observation is that in this last period, all methods (including the RND method based on a random selection of vertices to be removed) produce quite similar results.



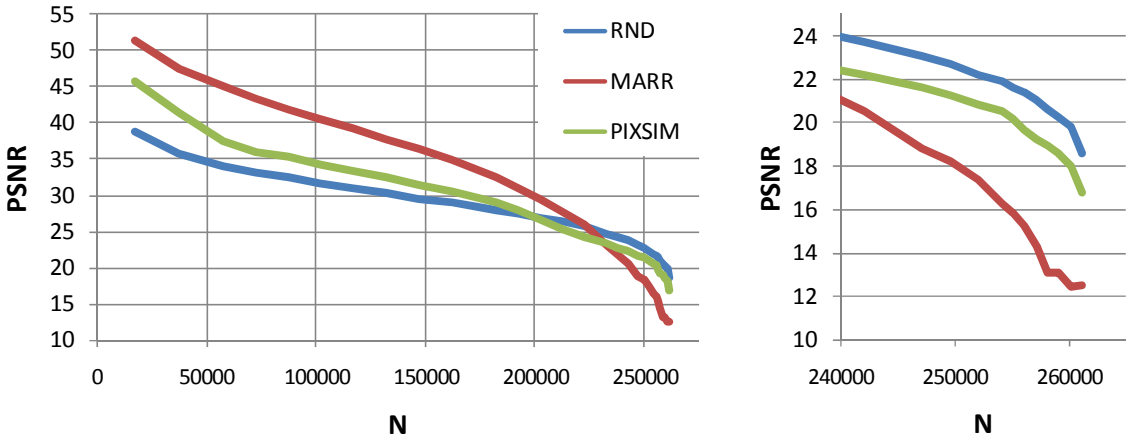
a) RND, PSNR = 23.53 b) MARR, PSNR = 19.53 c) PIXSIM, PSNR = 23.20

d) DISTW, PSNR=26.09 e) ERRDIST, PSNR=25.64 f) TRIMSE, PSNR = 27.82

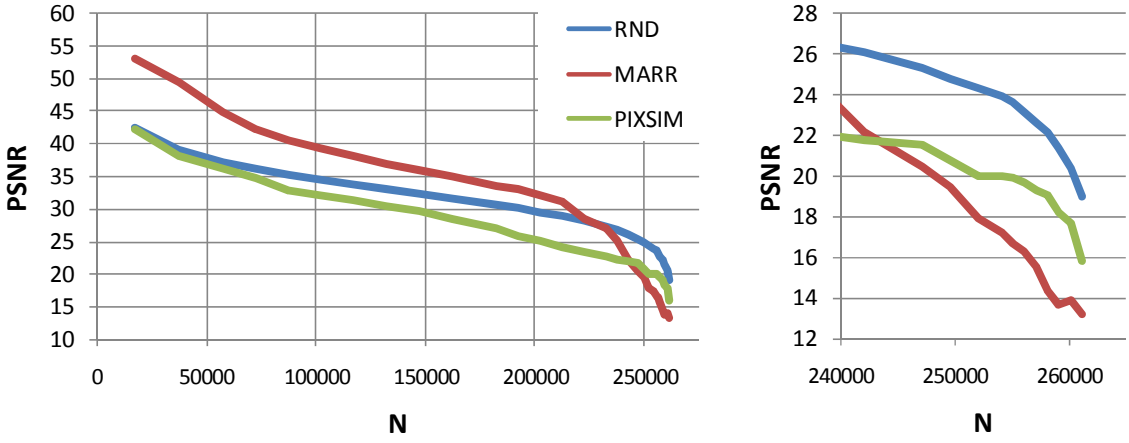
g) BRUTE, PSNR=32.08 h) NOCK, PSNR=30.58 i) GAUSS, PSNR = 31.94

Figure 4.16: *The comparison of fruit images reconstructed from triangulations with 10 000 vertices computed by various methods. Note that for the NOCK method, the triangulation had 10 134 vertices.*

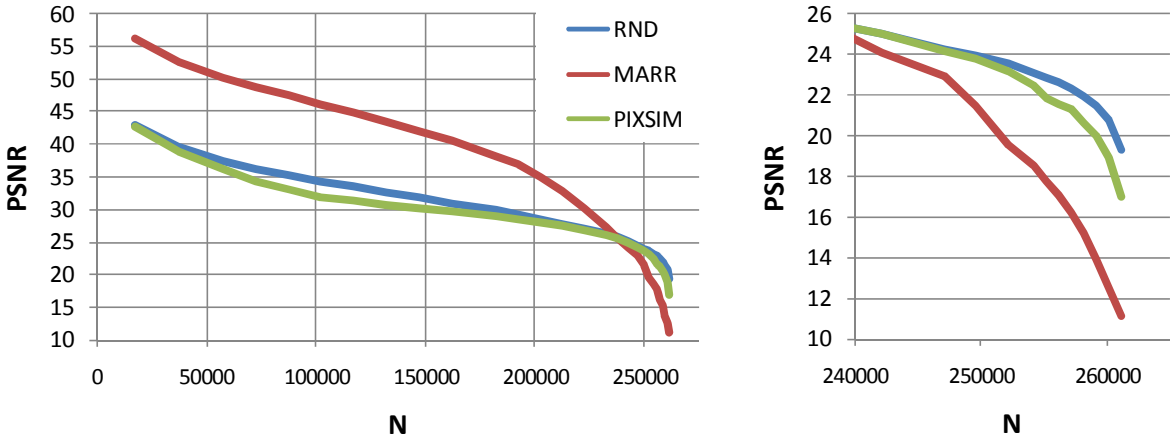
It means that an application that calls for a triangulation with a few vertices only (e.g., in non-photorealistic rendering), does not need to pay much attention which method for the evaluation of vertex significance to use. Another important observation is that a heuristics that produces results of a poor quality for triangulations of medium size may overcome other more sophisticated (and, therefore, also slower) methods when small triangulations are considered. For instance compare the behaviour of the RND and MARR heuristics (Figure 4.17).



a) boat (512×512)

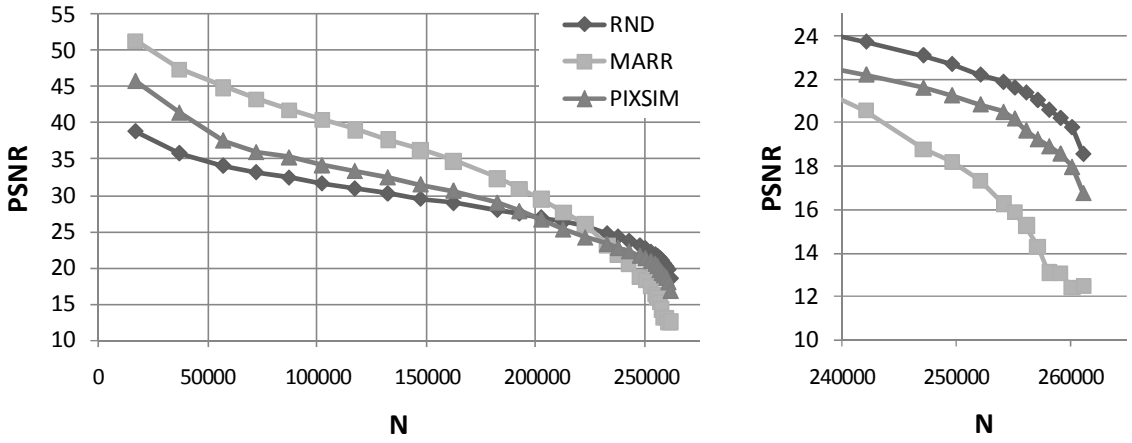


b) Lena (512×512)

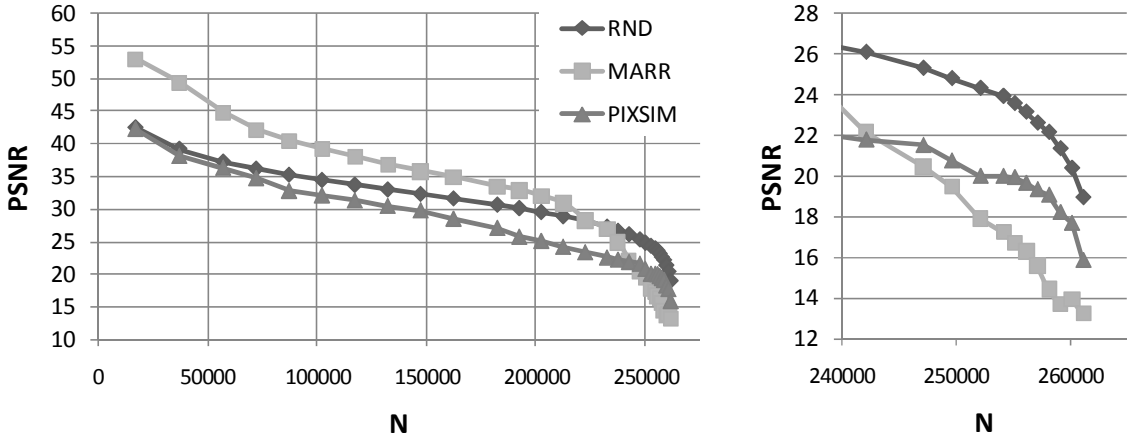


c) fruits (512×512)

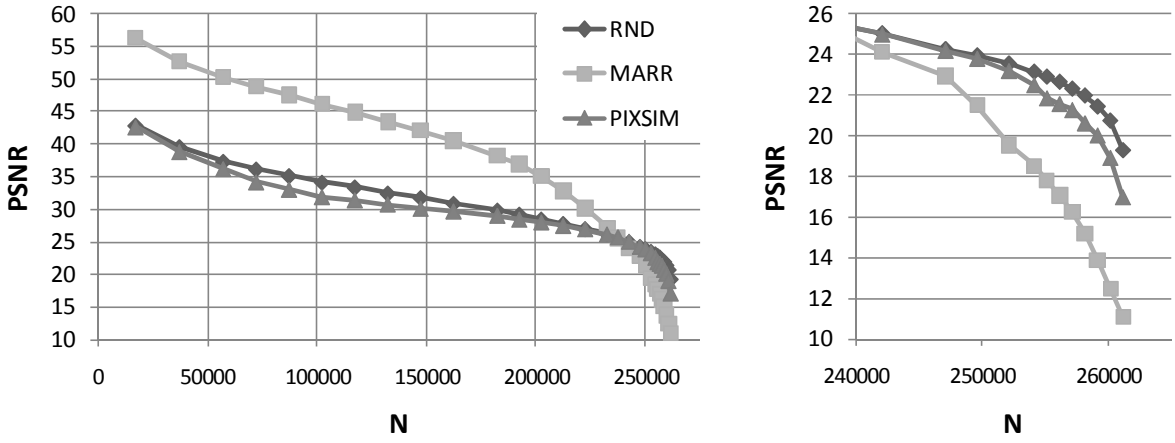
Figure 4.17 (colour): *The dependency of quality (measured in PSNR) of images reconstructed from Delaunay triangulations obtained by meshless heuristics on the number of removed vertices (higher values mean smaller triangulations) for three popular 512x512 grey-scale images.*



a) boat (512×512)

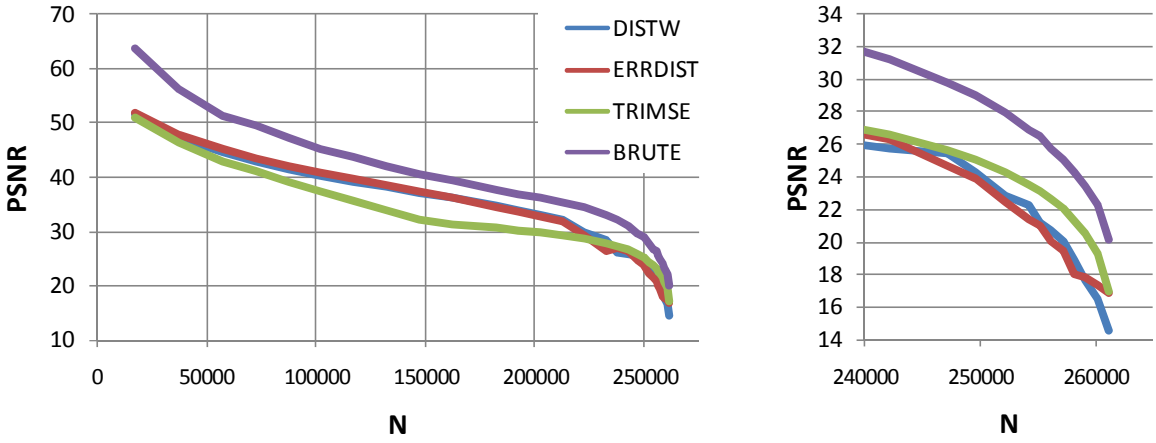


b) Lena (512×512)

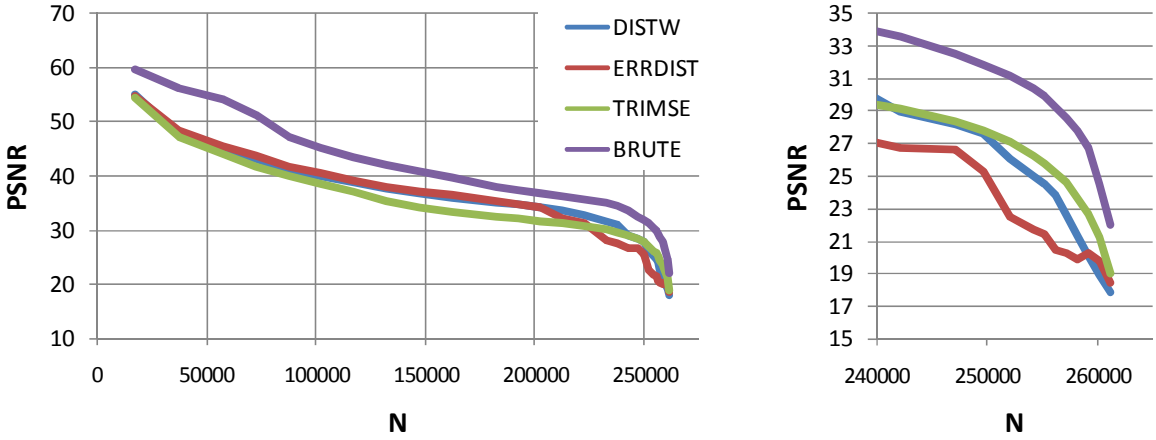


c) fruits (512×512)

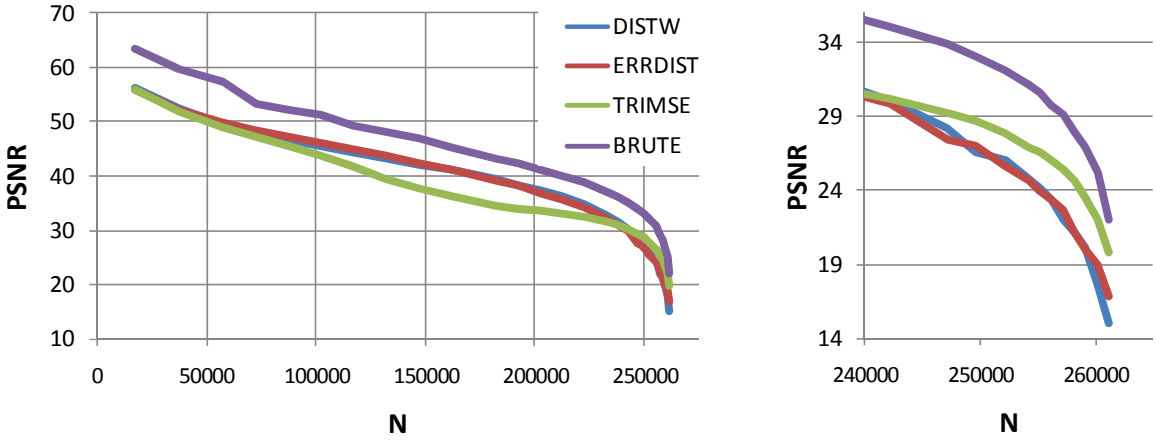
Figure 4.17: *The dependency of quality (measured in PSNR) of images reconstructed from Delaunay triangulations obtained by meshless heuristics on the number of removed vertices (higher values mean smaller triangulations) for three popular 512x512 grey-scale images.*



a) boat (512×512)

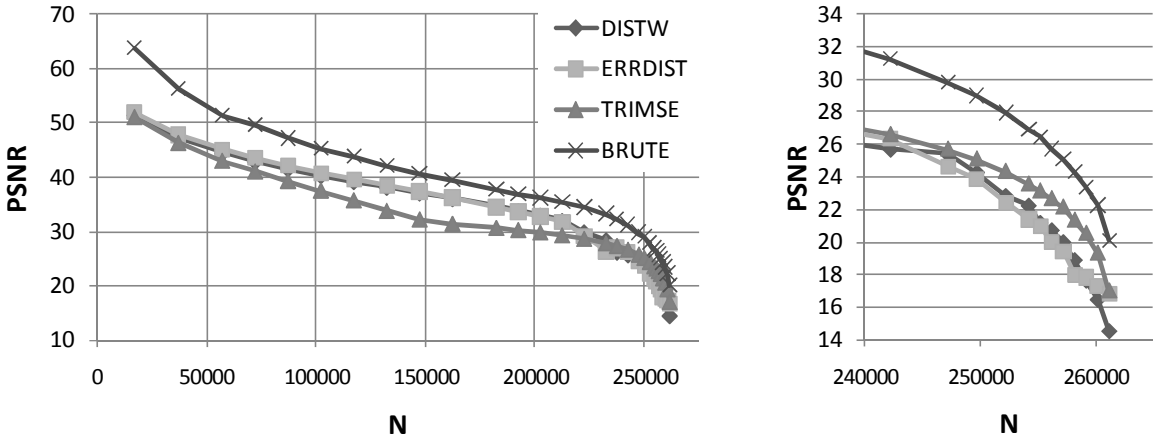


b) Lena (512×512)

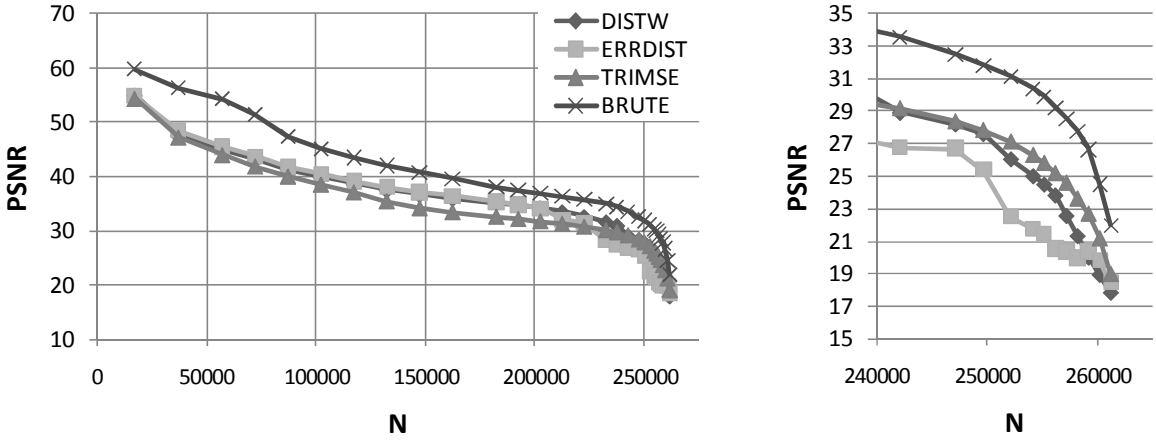


c) fruits (512×512)

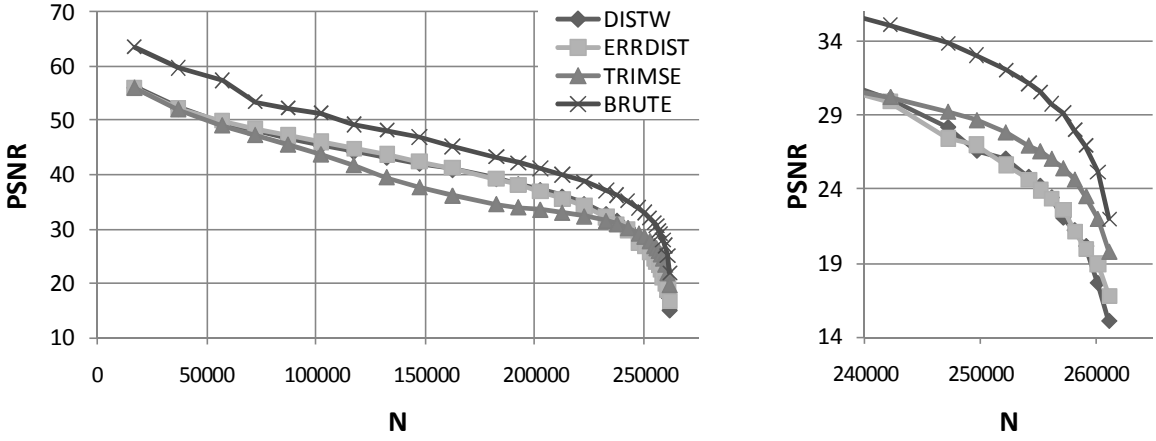
Figure 4.18 (colour): *The dependency of quality (measured in PSNR) of images reconstructed from Delaunay triangulations obtained by basic mesh based heuristics on the number of removed vertices (higher values mean smaller triangulations) for three popular 512x512 grey-scale images.*



a) boat (512×512)

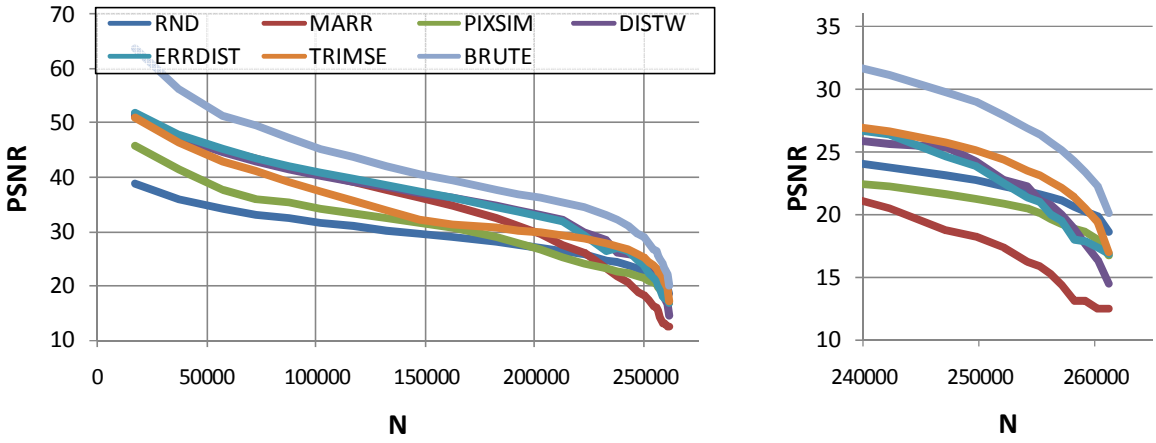


b) Lena (512×512)

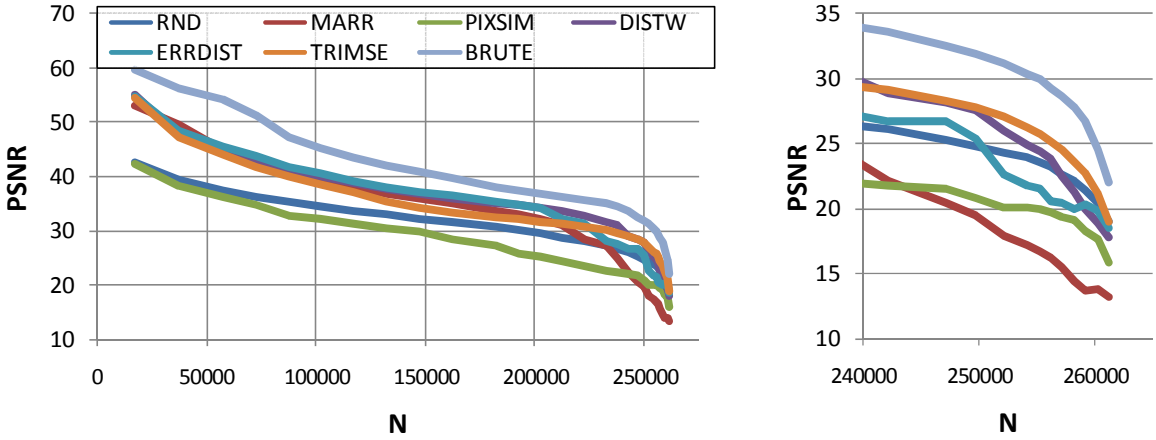


c) fruits (512×512)

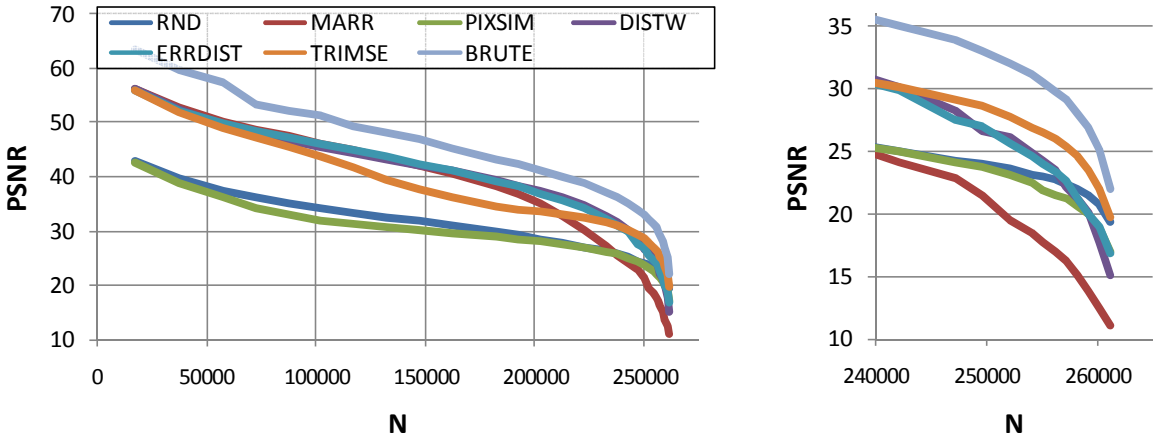
Figure 4.18: *The dependency of quality (measured in PSNR) of images reconstructed from Delaunay triangulations obtained by basic mesh based heuristics on the number of removed vertices (higher values mean smaller triangulations) for three popular 512x512 grey-scale images.*



a) boat (512×512)

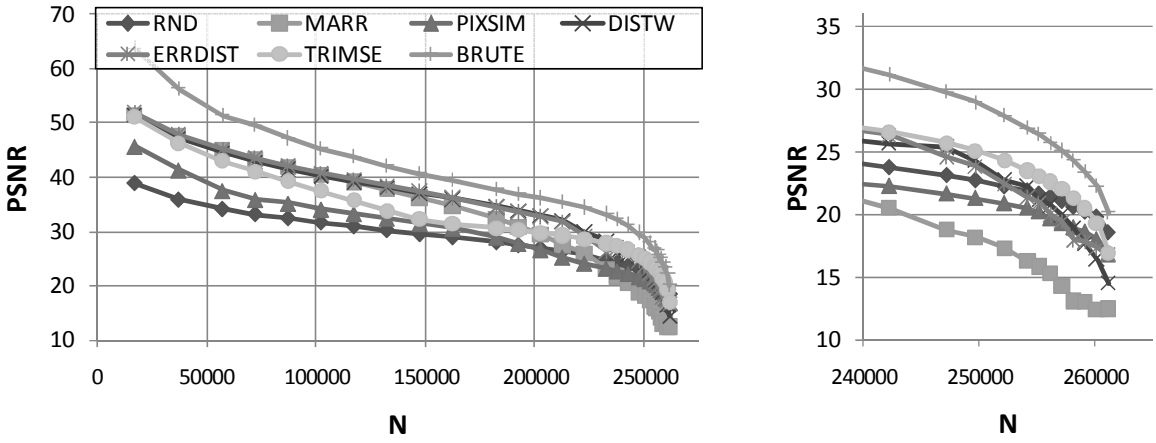


b) Lena (512×512)

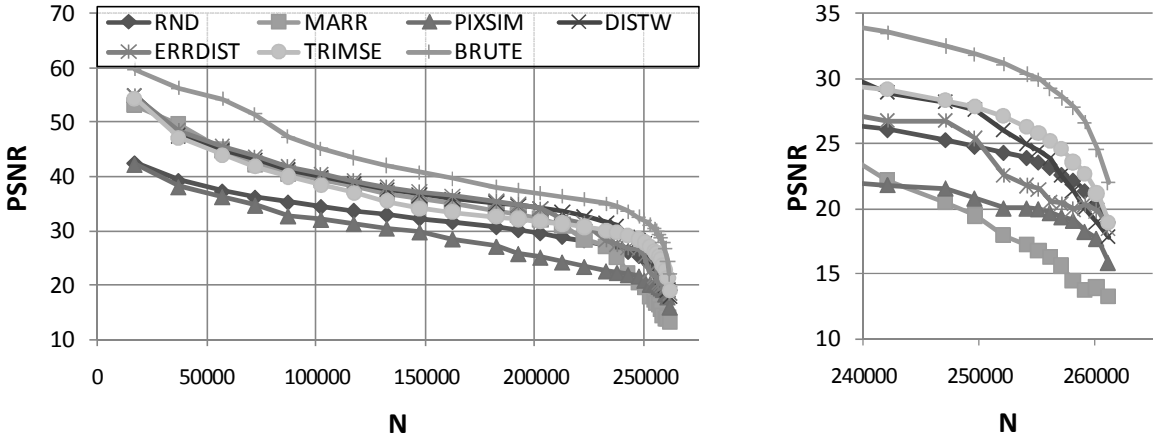


c) fruits (512×512)

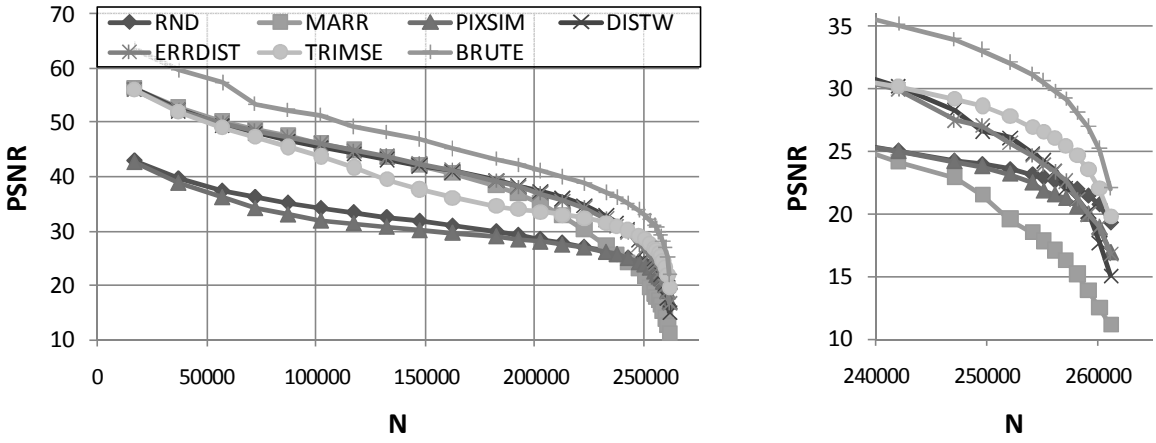
Figure 4.19 (colour): *The dependency of quality (measured in PSNR) of images reconstructed from Delaunay triangulations obtained by basic heuristics on the number of removed vertices (higher values mean smaller triangulations) for three popular 512x512 grey-scale images.*



a) boat (512×512)

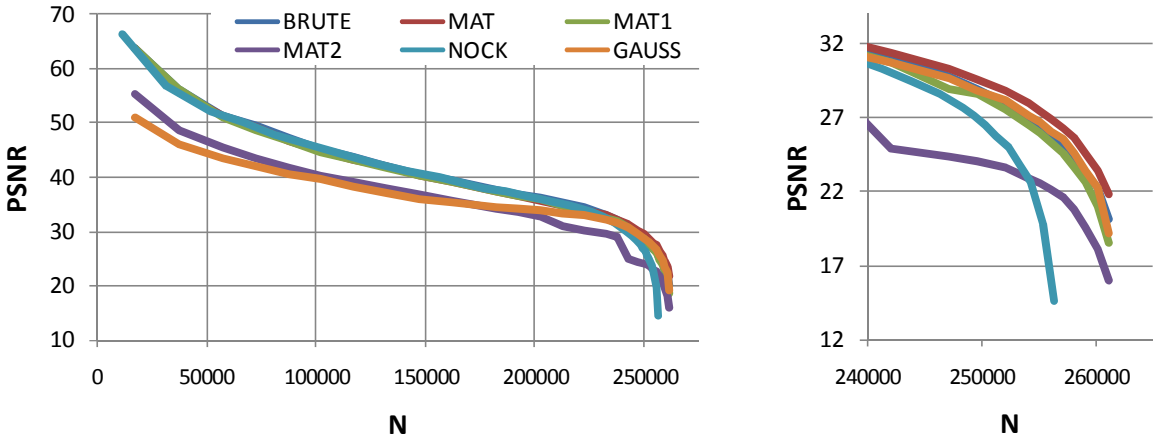


b) Lena (512×512)

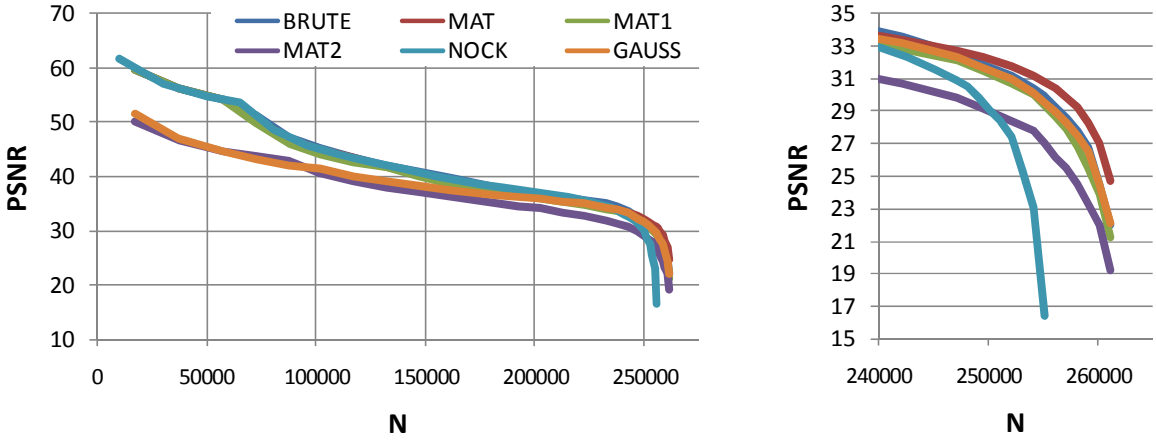


c) fruits (512×512)

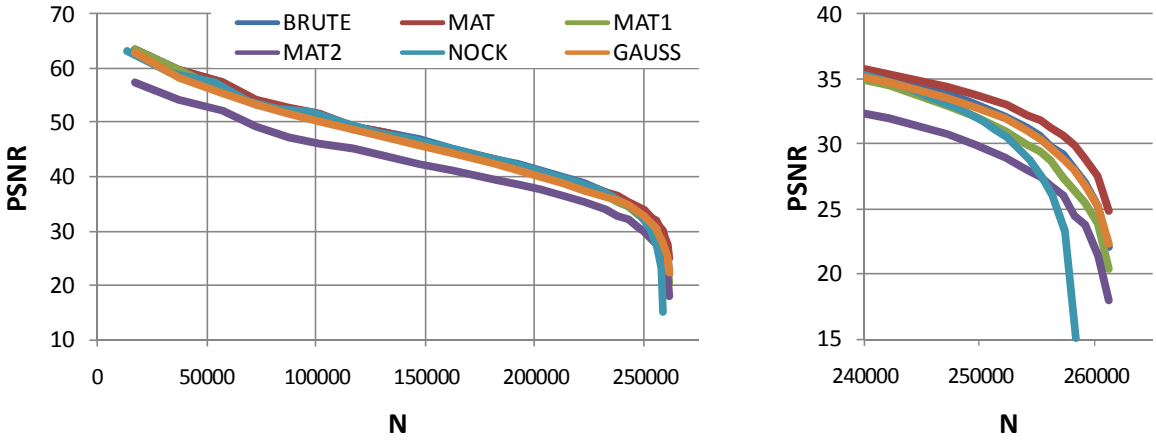
Figure 4.19: *The dependency of quality (measured in PSNR) of images reconstructed from Delaunay triangulations obtained by basic heuristics on the number of removed vertices (higher values mean smaller triangulations) for three popular 512x512 grey-scale images.*



a) boat (512×512)

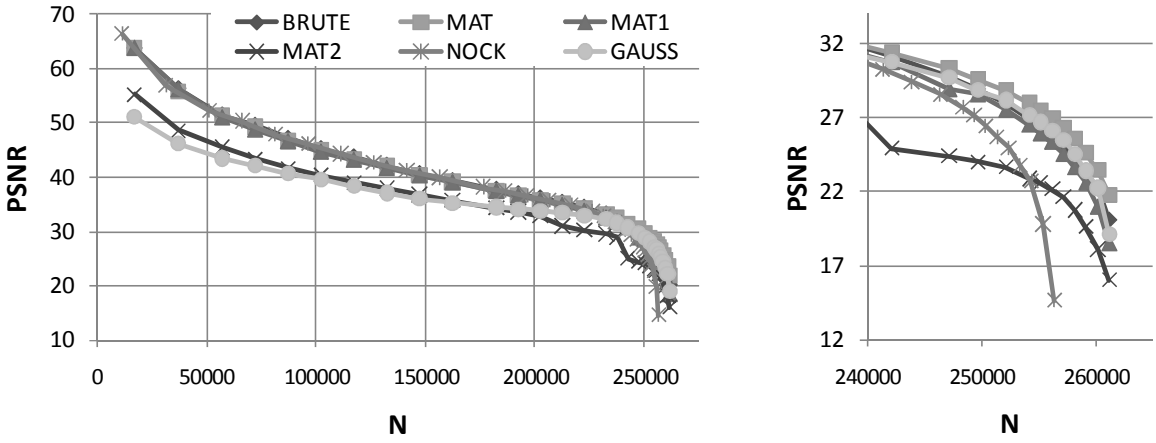


b) Lena (512×512)

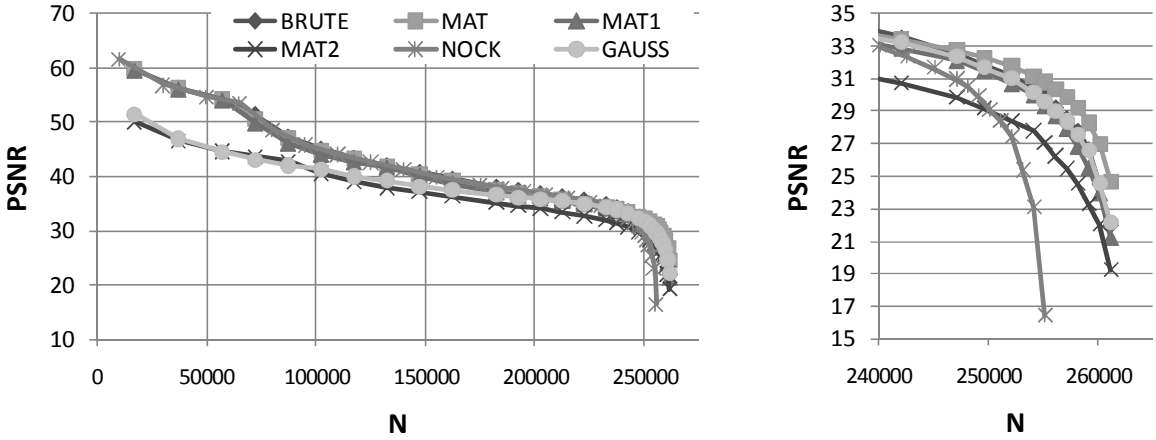


c) fruits (512×512)

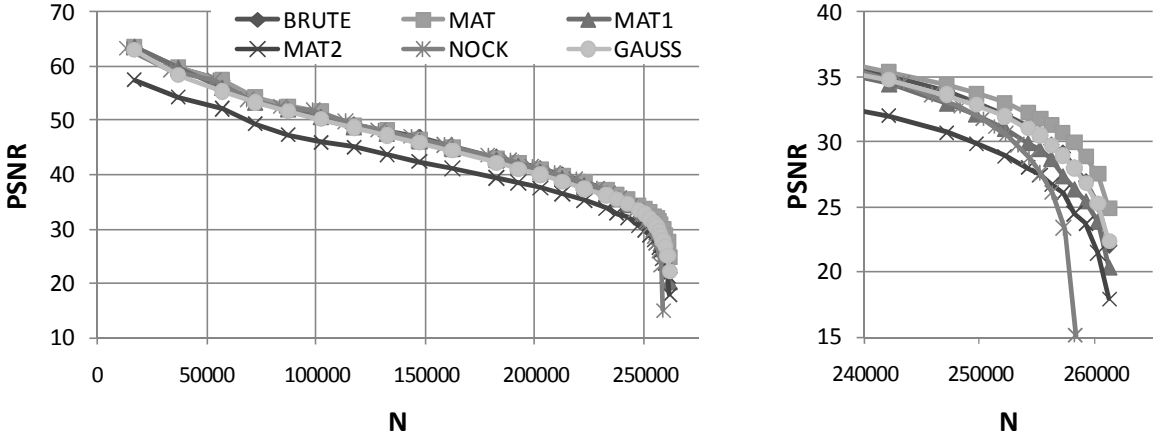
Figure 4.20 (colour): *The dependency of quality (measured in PSNR) of images reconstructed from Delaunay triangulations obtained by various BRUTE kind heuristics on the number of removed vertices (higher values mean smaller triangulations) for three popular 512x512 grey-scale images.*



a) boat (512×512)



b) Lena (512×512)



c) fruits (512×512)

Figure 4.20: *The dependency of quality (measured in PSNR) of images reconstructed from Delaunay triangulations obtained by various BRUTE kind heuristics on the number of removed vertices (higher values mean smaller triangulations) for three popular 512x512 grey-scale images.*

As it can be further seen in Figure 4.19, the curve of quality achieved by the slowest BRUTE method is the upper bound for all other curves, while the lower bound is formed by the curve achieved by the simplest (and the fastest as well) RND method in the area of large and medium triangulations and by another meshless heuristics, the MARR method, in the area of smaller triangulations.

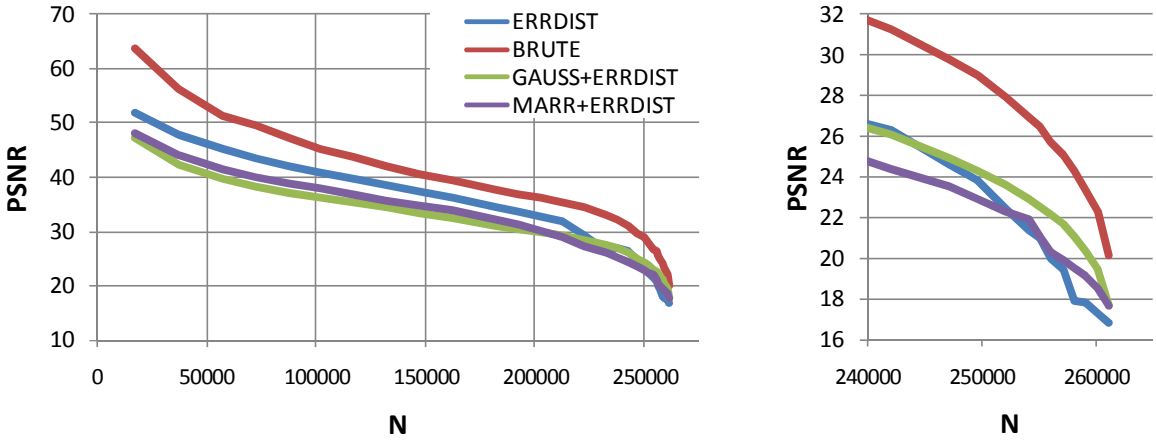
Except for the NOCK method and the BRUTE method in its MAT2 variant, whose behave poorly, there is no significant difference between various methods based on the brute-force strategy in the area of small triangulations – see Figure 4.20. Let us note that although the MAT variant, which was originally proposed by Demaret et. al [Dem06], proved to produce results of the highest quality, we continue to use the basic version of the BRUTE method in the further text, if not specified explicitly otherwise. Anyway, the basic version achieves only slightly worse results in comparison with the MAT variant.

4.4 Combined Heuristics

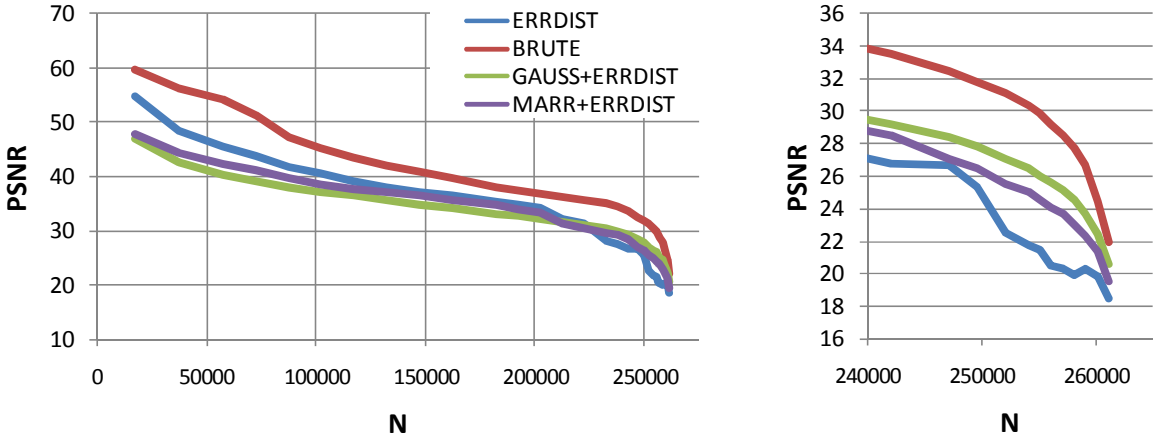
Figure 4.21 brings the comparison of results obtained by two combined methods with results that were obtained by their pure (not combined) counterparts. As it can be seen, when the initial significance of points is calculated by slow GAUSS technique and after that the points are processed by fast ERRDIST method, results achieved in the area of smaller triangulations are much better than when the pure ERRDIST method is considered. Although they do not reach qualities of the BRUTE method, this combined method is still very important for applications that need to convert quickly large images because it can process these images much faster than the BRUTE method whilst reaching acceptable qualities. The reason of this is that the time complexity of the BRUTE method grows much faster than the time complexity of the significance evaluation used in the GAUSS method.

The other combined method, where the initial significance of points is evaluated using the Marr-Hildreth operator, does not outperform the previous one and it might be even worse than the ERRDIST method itself in some cases (see the results for the boat image). Nevertheless, this combined method has also its merits as it is much faster than the previous one and, therefore, it can be used as a good compromise between speed and quality. It is important, however, to point out that this method is pretty useless when dealing with larger triangulations.

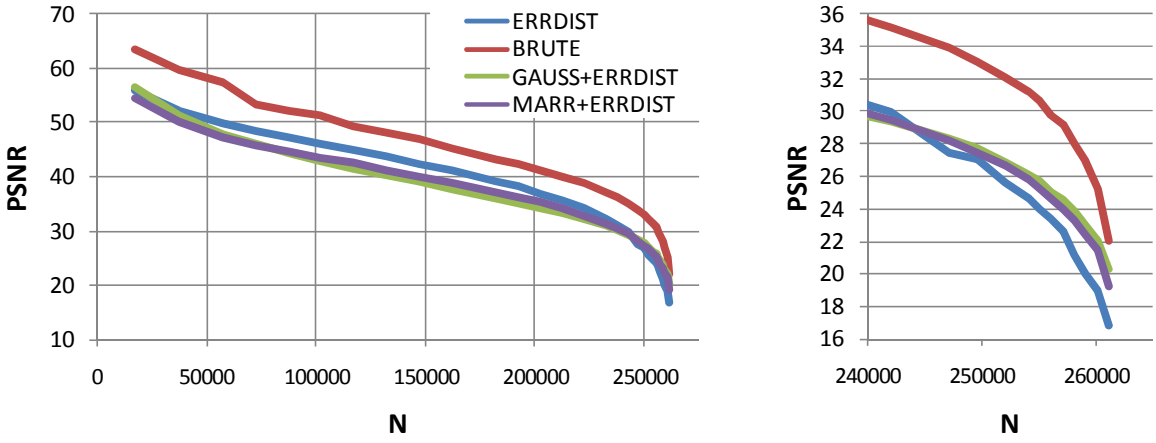
We also experimented with other combinations (e.g., the MARR combined with the basic BRUTE method) but we did not find any other combination that would bring a substantial improvement. An interesting observation from our results is that the decimation technique always produces a set of significant points where the majority is formed by points that belong to image edges no matter which mesh based heuristics is used. However, when we try to enforce edge points by giving them larger initial significance, the results are typically worse than without this interfering.



a) boat (512×512)

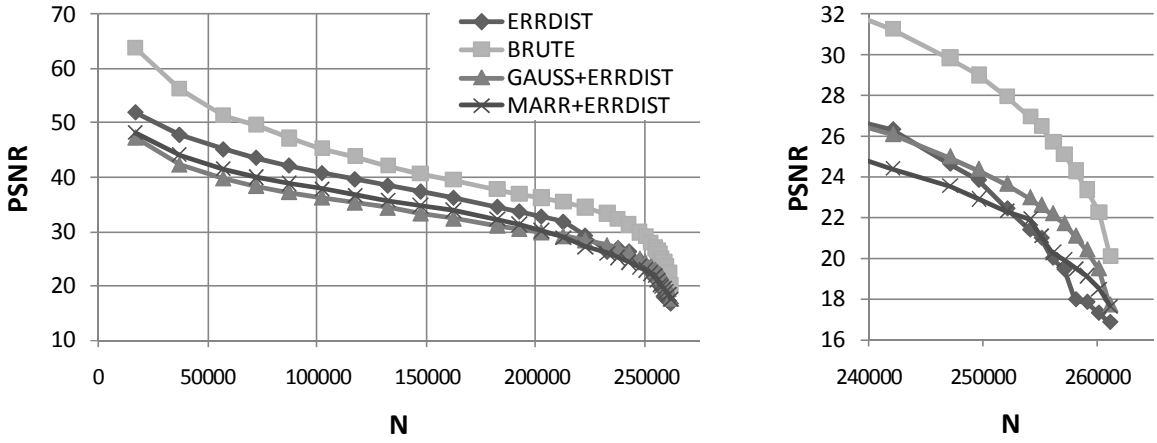


b) Lena (512×512)

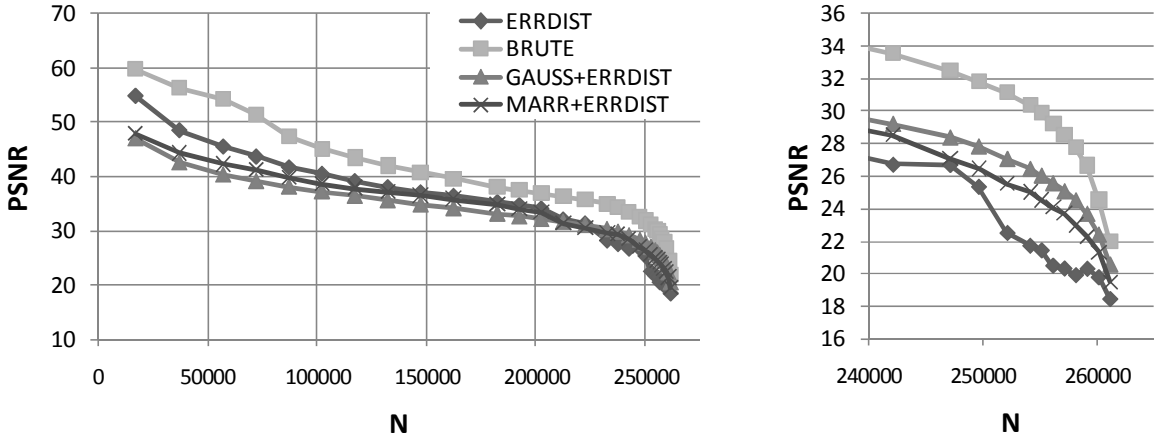


c) fruits (512×512)

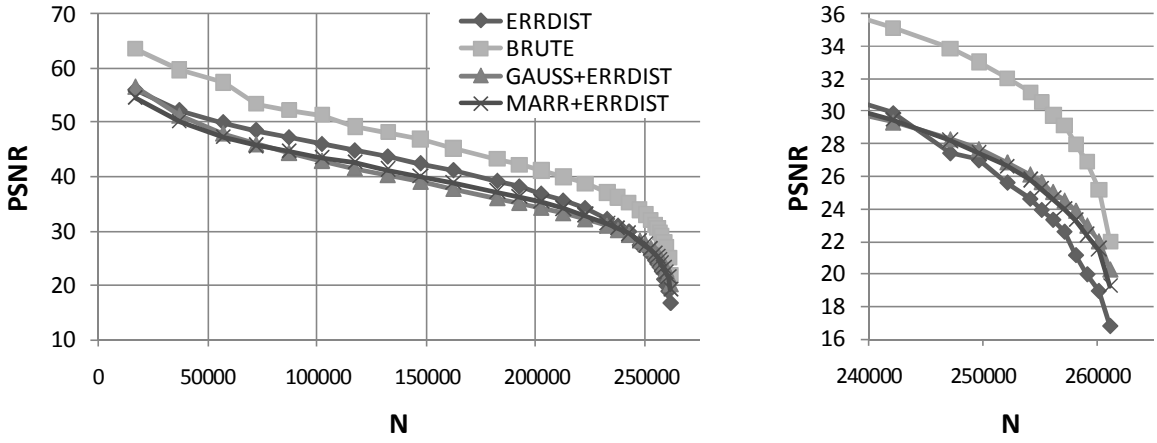
Figure 4.21 (colour): *The dependency of quality (measured in PSNR) of images reconstructed from Delaunay triangulations obtained by the ERRDIST combined heuristics on the number of removed vertices (higher values mean smaller triangulations) for three popular 512x512 grey-scale images. The results are compared with the results of ERRDIST and BRUTE methods.*



a) boat (512×512)



b) Lena (512×512)



c) fruits (512×512)

Figure 4.21: *The dependency of quality (measured in PSNR) of images reconstructed from Delaunay triangulations obtained by the GAUSS combined with ERRDIST heuristics on the number of removed vertices (higher values mean smaller triangulations) for three popular 512x512 grey-scale images. The results are compared with the results of ERRDIST and BRUTE methods.*

4.5 Combined Triangulation Strategy

In order to improve our results, we decided to mix the decimation technique with the refinement strategy as follows. The algorithm keeps two priority queues, denoted as Q_P and Q_R. The first queue (Q_P) contains vertices currently present in the Delaunay triangulation and the other contains vertices removed from the triangulation in previous iterations of the algorithm. Whilst the point at the head of the first queue is the least significant one, the point at the head of the latter queue is the one with the highest significance of all points in this queue. In iteration, the algorithm compares significances of both points at heads of queues. If the point from the queue Q_R is more significant than the other one, then with the probability p, this point is inserted into the triangulation; otherwise the point from at the head of the queue Q_P is removed from the triangulation. In any case significances of points (both present and already removed) in the affected area must be recalculated. The probability p is proportional to the number of already removed points, i.e., it is more probable for the removed vertex that it will be reinserted into the triangulation in later stages of the algorithm.

Without any doubt, the proposed algorithm consumes more time than the one based on the decimation technique and with an improper probability function it may even ends in an infinite loop when some points are removed to be inserted after a while. We insert points when the following condition is fulfilled:

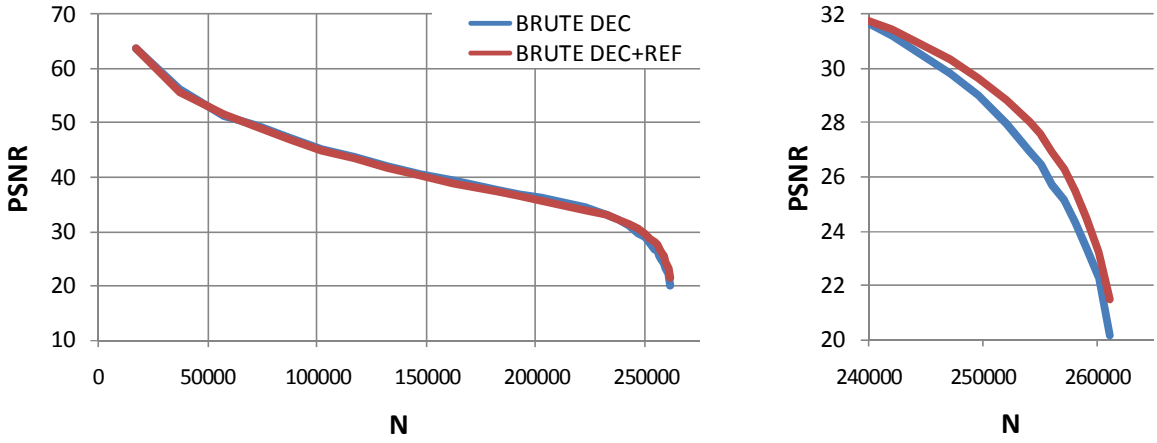
$$count(Q_R) \cdot rnd() > \frac{4}{5} N \cdot M,$$

where $count(Q_R)$ is the number of points in the queue Q_R, *rnd*() is a random function that returns real numbers from the interval <0, 1> and N and M are width and height of the original raster image. With this condition the algorithm terminates eventually, however, it is still several times slower than the decimation.

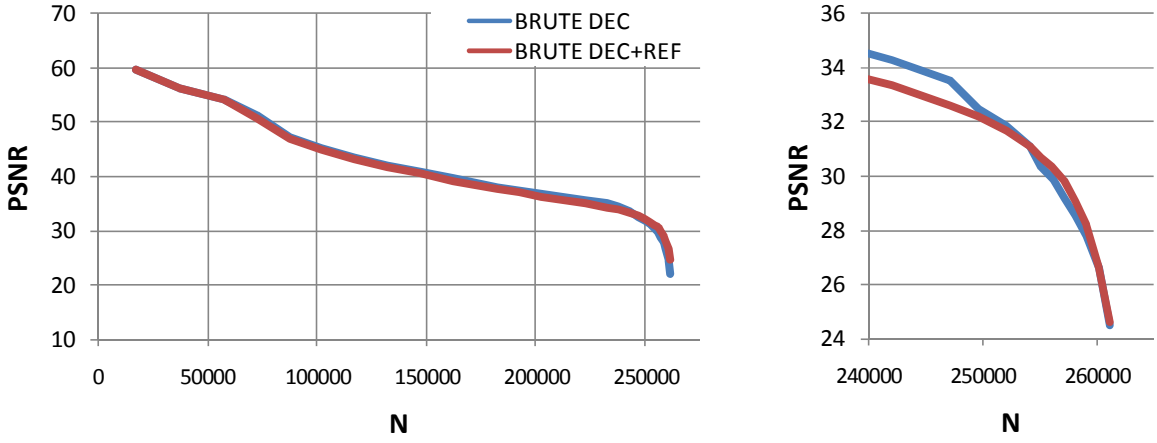
Figure 4.22 compares the results obtained by the BRUTE heuristics (which has proven to be the best one) when the decimation technique was used only and when the decimation was mixed with the refinement strategy. Clearly the combined algorithm achieves better results in almost all cases. The improvement is, however, not significant and if we consider the additional time requirement for this algorithm, we have to prefer the decimation algorithm.

4.6 Image Filtering

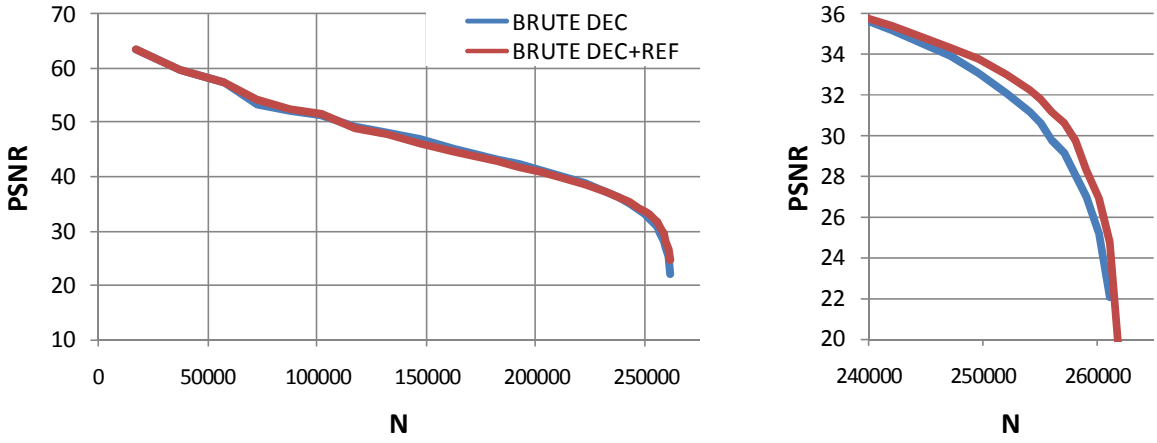
Another option how to improve the results is to apply some lossless filtering technique that transforms the input data into a form more suitable for the considered heuristics. For an easier understanding of the problem, let us resort to one dimensional case. Figure 4.23 shows a function that should be approximated by a piecewise linear function with the allowed approximation error ε. In its original form, an approximation that connects the ending points of the given function, p_a and p_b, is not possible because its error is out of the specified tolerance. It is, therefore, necessary to introduce the third vertex, p_c, into the approximation to fulfil the criterion. If the input function is, however, filtered using a simple SUB filter (will be described later), the approximation by p_a and p_b is possible. It means that by the filtering we reduced the number of vertices from three to two, i.e., we either improved the compression ratio or spared one vertex that can be used elsewhere to improve the quality of the geometric representation.



a) boat (512×512)

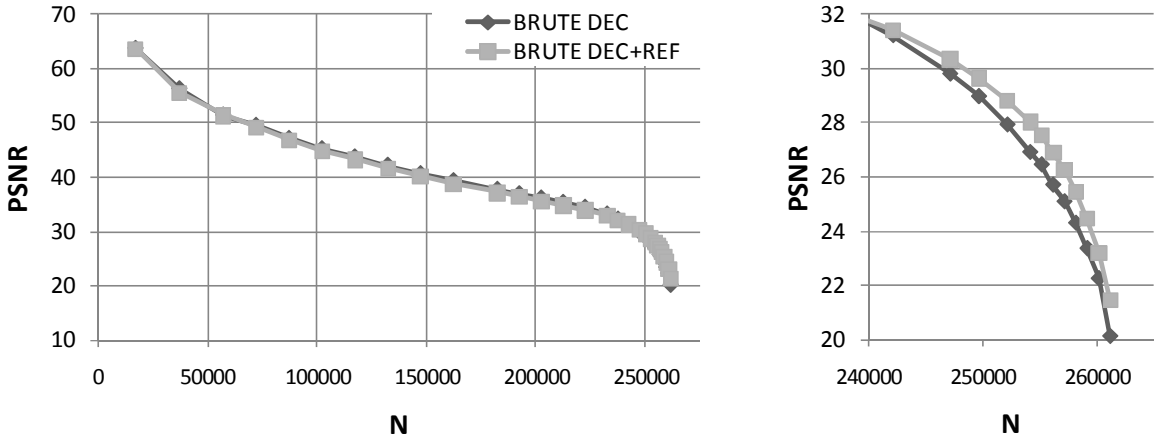


b) Lena (512×512)

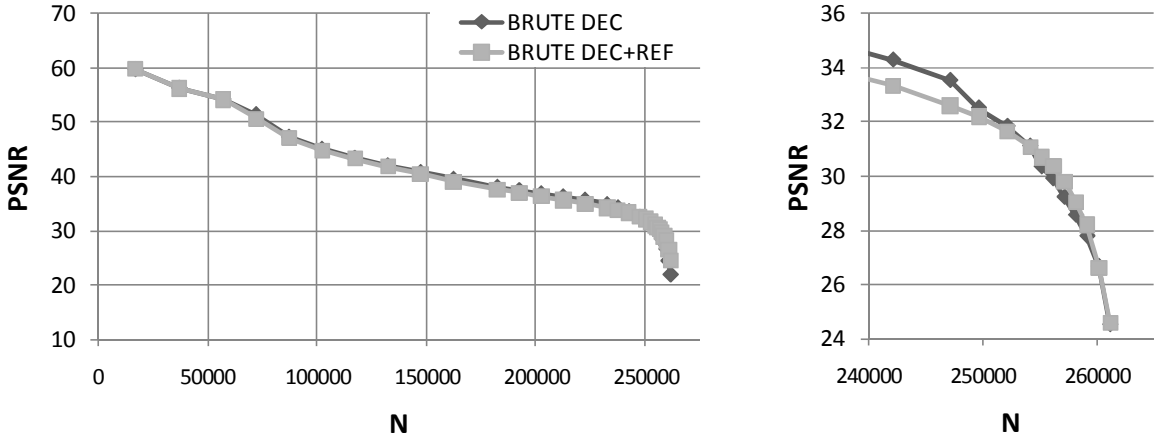


c) fruits (512×512)

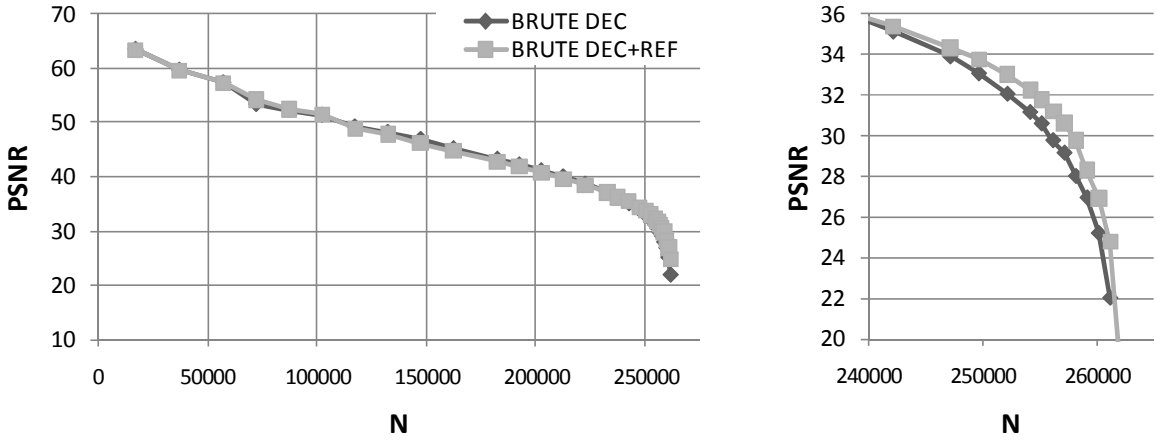
Figure 4.22 (colour): *The dependency of quality (measured in PSNR) of images reconstructed from Delaunay triangulations obtained by the BRUTE heuristics with (DEC+REF) and without (DEC) refinement strategy on the number of removed vertices (higher values mean smaller triangulations) for three popular 512x512 grey-scale images.*



a) boat (512×512)

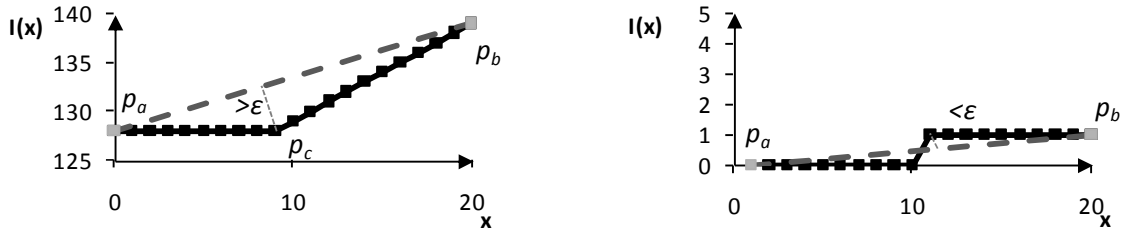


b) Lena (512×512)



c) fruits (512×512)

Figure 4.22: *The dependency of quality (measured in PSNR) of images reconstructed from Delaunay triangulations obtained by the BRUTE heuristics with (DEC+REF) and without (DEC) refinement strategy on the number of removed vertices (higher values mean smaller triangulations) for three popular 512x512 grey-scale images.*



a) the original function b) the filtered function

Figure 4.23: *An edge approximation (dashed line) of the original and filtered function (solid line).*

We experimented with three image filters commonly used in PNG format. The first one is already mentioned SUB filter that computes differences between neighbouring pixels:

$$SUB(x) = mod(I(x) - I(x-1)),$$

where x ranges from zero to the number of pixels in the image minus one, $I(x)$ refers to the grey value of the pixel in the image corresponding to the specified position x and $mod(x)$ denotes unsigned arithmetic modulo 256, so the outputs fit into bytes (e.g., 1 – 2 = 255). For all negative x, we assume $I(x) = 0$. In order to reverse the effect of the SUB filter after the interpolation of triangles, the output is computed simply as:

$$I(x) = mod(SUB(x) + (x-1)).$$

The AVG filter transmits the difference between the value of a pixel and the average of the two neighbouring pixels (left and above) used as a prediction of this value. The formulas for forward and reverse filter can be written as:

$$AVG(x) = mod\left(I(x) - \frac{I(x-1)+I(x-N)}{2}\right),$$

$$I(x) = mod\left(AVG(x) + \frac{I(x-1)+I(x-N)}{2}\right),$$

where N denotes the horizontal size of the image.

As the previous filter, the PAETH filter also transmits the difference between the real value and the predicted value of a pixel. The prediction is, however, calculated from the three neighbouring pixels (left, above, upper left) by the algorithm developed by Alan W. Paeth. This pseudocode of this algorithm is depicted in Figure 4.24.

```
p = a + b – c;   pa = abs(p - a);   pb = abs(p - b);   pc = abs(p - c);
if pa <= pb and pa <= pc then PAETH(x) = a;
else if pb <= pc then PAETH(x) = b;
else PAETH(x) = c;
```

Figure 4.24: *PAETH filter (see http://www.w3.org/TR/PNG/).Legend: a is the left pixel value, b is the grey value of pixel above and c is the upper left pixel.*

Despite our expectations, the experiments proved that these filtering techniques are not useful; we obtained even worse results with them than without. Figure 4.25 shows images of fruits that were reconstructed from triangulations with 93.4% of the original amount of vertices (i.e., only an insignificant amount of vertices was removed) when filtering techniques were applied. Artefacts are clearly visible.

a) SUB b) AVG c) PAETH

Figure 4.25: *Artefacts caused by various filtering techniques.*

We identified several reasons for such behaviour. The most important fact is that by filtering we introduce a dependency between pixels and, therefore, if one pixel is reconstructed with an error, this error is distributed over the rest of pixels, which may cause unexpected artefacts. Let us consider the following example. The SUB filter transforms a group of adjacent pixels 0, 0, 10, 10 and 10 into filtered values 0, 0, 10, 0 and 0. If the second value is not stored and has to be reconstructed, we get values 0, **5**, 10, 0, 0. The reverse SUB filter propagates the error and gives pixels 0, **5**, **15**, **15** and **15**.

The problem is also that although the filtering flattens the image, it does not create sufficiently large places with a constant value but, on the contrary, it introduces a lot of edges into the image. It makes the approximation process uneasy as it leads to a rapid degradation of quality. Actually, this is close to the MARR method.

4.7 Summarization

Let us summarize what we have learned from experiments presented in this section. In order to get a geometric representation with an acceptable quality, the evaluation of significant points and their triangulation must be two related, not separable, steps. Any mesh based heuristics described in the previous section is suitable when used with the decimation strategy. The best (in the achieved quality of representation) is apparently the BRUTE method. This method is, however, quite slow and, therefore, if an application calls for fast transformation, the ERRDIST or DISTW methods are often optimal. The majority of points in the produced representation are points that represent edges in the original raster image, so one can be tempted to assign higher initial significance to every point on these edges in order to help the heuristics to get better results. This strategy, unfortunately, does not work well as many points in the original image lie on edges. Apparently, the better results could be achieved, if only some of those points were set to be more significant. How to detect them is, however, an issue. Filtering of input image (by filters used in PNG), which was supposed to help to get triangulations with fewer vertices, also proved to be useless. Nevertheless, we believe that the idea of filtering of image in the pre-processing is not bad in general but one has to come with the filter where the filtered values are more independent and thus less liable to errors. Another way how to improve results is to exploit a different interpolation of triangles. This option is discussed in Section 7.

5 Triangulation Encoding

Having an arbitrary triangulation representing the image, it is necessary to store both, the coordinates and grey values of its vertices (i.e., the geometry) and triangles (i.e., the topology). If the image is, however, represented by the Delaunay triangulation, it is possible to avoid storing of the topology because, as the Delaunay triangulation of a given vertex set is unique (if no four points lie on a common circum-circle), it may be recomputed from the geometry during the reconstruction process. At any rate, even if it contains a few vertices only, the triangulation in this raw form consumes a lot of bytes and, therefore, it is not suitable for storing. A more compact form is necessary.

In this section, we describe various triangulation encoding methods. In order to get as small files as possible, we often employ also some of existing data compression algorithms such as bzip2, deflate (the default algorithm used in ZIP), ppdm, lzma (both used in 7z), paq8o, lpaq1, and quad – see [Mah07, Sou07b, Wik07a, Wik07b, Wik07c]. Let us note that we exploited the implementation of these algorithms that is in PeaZip utilities [Sou07a]. Commands used to compress the given source file using these utilities are shown in Table 5.1

Algorithm	*Command*
bzip2	7z.exe a -tbzip2 -mmt=on -mx9 -md=900k -mpass=5
deflate	7z.exe a -tzip -mm=Deflate64 -mx9 -md=64k -mfb=128 -mpass=5 -mem=AES256
ppdm	7z.exe a -t7z -m0=PPMd -mx9 -mmem=192m -mo=32 -ms=on
lzma	7z.exe a -t7z -m0=LZMA -mmt=on -mx9 -md=32m -mfb=64 -ms=on
paq8o	paq8o.exe -1
lpaq1	lpaq1.exe 6
quad	quad.exe -x

Table 5.1: *Commands used for the compression of a source file.*

5.1 Raw

After a header, which contains the size of image and the number of vertices, is written into the output file, the method proceeds with storing of vertices one after another in an uncompressed way. For each vertex, it saves its x and y coordinates followed by its grey value using 16-bit integer for one coordinate and 8-bit for the grey value, i.e., 5 bytes per one vertex are consumed. The topology is not stored at all, i.e., this method is suitable for the Delaunay triangulation only. The produced file is afterwards compressed by one of data compression algorithms. As general compression algorithms usually do not take the character of data into an account (except for paq8o), the expected compression ratio is rather small.

5.2 Vertex Path (VXPATH)

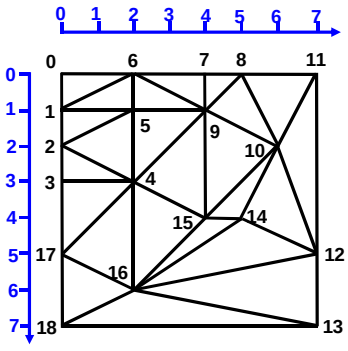
The VXPATH method successively visits all vertices storing the differences (in both coordinates as well as in grey values) between the currently inspected vertex and the previously visited vertex into two arrays constructed in the memory. In the first array, denoted as *V*, there are stored differences in x and y-coordinates; the second, denoted as *C*, keeps differences in

grey values (i.e., in z-coordinates). In the reconstruction process, vertex positions are then reconstructed from the array *V*, whilst the second array *C* is used for the computation of grey values.

Vertices are visited in such an order that the differences in x and y-coordinates are minimized. Being in the vertex $p(p_x, p_y, p_{grey})$, the algorithm thus proceeds with such vertex $q(q_x, q_y, q_{grey})$ that it has not been already visited and the Minkowski distance between these vertices [Wik07d], i.e., the value:

$$|p_x - q_x| + |p_y - q_y|,$$

is minimal. Figure 5.1 shows the triangulation representing a tiny image of 8x8 pixels with the displayed order of vertices, their grey values and the corresponding content of both arrays.



Grey values

# grey		# grey	
0	4	10	9
1	2	11	15
2	5	12	18
3	5	13	20
4	7	14	12
5	6	15	9
6	4	16	4
7	10	17	3
8	12	18	0
9	7		

V (Differences in x, y)

x	y	x	y	x	y	x	y	x	y	x	y	x	y	x	y	x	y	x	y
0		1		2		3		4		5		6		7		8		9	
0	0	0	1	0	1	0	1	2	0	0	-2	0	-1	2	0	1	0	-1	1

10		11		12		13		14		15		16		17		18	
2	1	1	-2	0	5	0	2	-2	-3	-1	0	-2	2	-2	-1	0	2

C (Differences in grey values)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
4	-2	3	0	2	-1	-2	6	2	-5	2	6	3	2	-8	-3	-5	-1	-3

Figure 5.1: *The storing order of vertices and the differences in both coordinates and grey values for VXPATH.*

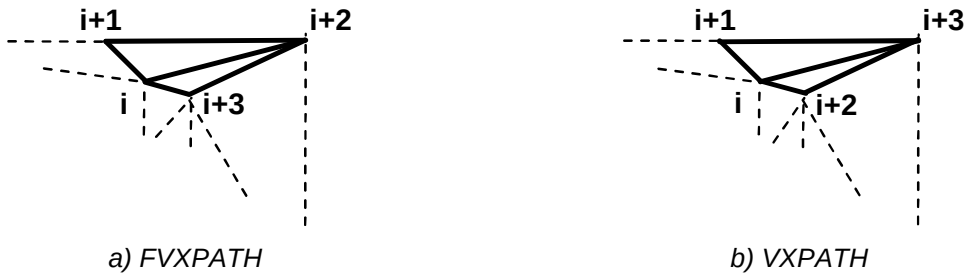
The method then stores the header (see the Raw method) and the minimum in the arrays of differences in x and y using 16-bits into the output file. Afterwards, all values from this array, lowered by the minimum, are stored using as small number of bits as possible (constant for every value). The number of used bits, naturally, is written into the file first. In our example, the minimum is -2 and the maximum is 7, i.e., the range is 9 and, therefore, we need 4 bits. Let us note that it is more than we would have required, if we had stored coordinates instead of differences (only 3 bits would have been required because the largest coordinate is 7). For larger images, however, it is highly improbably that the storing of differences would consume more bits per vertex than the storing of coordinates themselves. Anyway, this approach lowers the data entropy and, therefore, if we use one of already mentioned data compression utilities on the output file, we can expect higher compression ratios.

Differences in grey values are stored otherwise. It is reasonable to expect that two adjacent vertices may hold a completely different grey value. This is definitely true for vertices in the vicinity of image edges where the difference can be even 255 in the worst case. If we had applied the same encoding strategy to this second array of differences, we would have needed, typically, with at least 8 bits per one value. Therefore, values are simply stored using 8 bits.

An advantage of this approach is that it does not require the connectivity for the decoding process and it offers a compactness as the differences should be very small. The output file can be optionally compressed even more by one of general data compression algorithms. On the other hand, this brute-force algorithm runs in $O(N^2)$, which means that it takes a lot of time (especially, if the triangulation contains a large number of vertices).

5.3 Faster Vertex Path (FVXPATH)

This method is a slight modification of the VXPATH method. Being in the vertex *p*, the algorithm computes its Minkowski distance from every vertex that has not been already visited and is connected with this vertex by an edge in the triangulation and it proceeds with the vertex for which the distance is minimal. If no such vertex exists, an exhaustive search is used like in VXPATH. The algorithm, therefore, runs much faster than VXPATH but it generates a slightly different array of differences that may contain larger values – see Figure 5.2. Let us note that for the tiny triangulation from Figure 5.1 both algorithms produce the same results. Unlike VXPATH, the algorithm encodes distances in x and y-coordinates separately, i.e., it constructs two arrays, V_x and V_y, instead of just one array *V*.



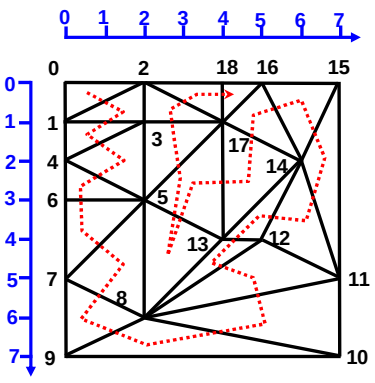
a) FVXPATH *b) VXPATH*

Figure 5.2: *The difference between the order of vertices in FVXPATH and VXPATH methods.*

As for the differences in grey values, we decided, despite the reason given in the previous section, to store them in the same way as the differences in coordinates, thus accepting that in the worst case this may lead to the nine-bits per vertex representation.

5.4 Triangle Path (TRPATH)

The Triangle Path enhances the VXPATH method in two small things. First, it the processes vertices in a different order (but by the same way, i.e., it also constructs arrays of differences) as follows. The algorithm traverses triangles in the triangulation in the depth-first order and whenever a new triangle is visited, its still not processed vertices are processed. An example is given in Figure 5.3.



Grey values

#	grey	#	grey
0	4	10	20
1	2	11	18
2	4	12	12
3	6	13	9
4	5	14	9
5	7	15	15
6	5	16	12
7	3	17	7
8	4	18	10
9	0		

V (Differences in x, y)

x	y	x	y	x	y	x	y	x	y	x	y	x	y	x	y	x	y	x	y
0		1		2		3		4		5		6		7		8		9	
0	0	0	1	2	-1	0	1	-2	1	2	1	-2	0	0	2	2	1	-2	1
10		11		12		13		14		15		16		17		18			
7	0	0	-2	-2	-1	-1	0	2	-2	1	-2	-2	0	-1	1	0	-1		

C (Differences in grey values)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
4	-2	2	2	-1	2	-2	-2	1	-4	20	-2	-6	-3	0	6	-3	-5	3

Figure 5.3: *The storing order of vertices and the differences in both coordinates and grey values for TRPATH. The traversal order of triangles is drawn by a dotted poly-line.*

As vertices are handled in a linear time, the time consumption is significantly reduced. On the other hand, the vertex *q* adjacent to the vertex *p* in the storing order is rarely the closest one in the meaning of Minkowski distance. Sometimes it may even happen that these vertices are quite far away from themselves and/or separated by several triangles. This is caused by dead ends in the traversal process – see the triangle [5, 13, 8] in Figure 5.3. All in all, it means that we can expect larger values in the array *V*.

The second improvement is, therefore, that this method does not use a fixed number of bits for all differences but encodes the array *V* using variant number of bits, i.e., different parts of the array are encoded using different numbers of bits. This makes the method slightly more complex but promises lower storage costs. Let us now describe the encoding in detail.

For every part, it is necessary to store the number of values present in this part (16 bits), their minimum (another 16 bits) and, indeed, the number of bits used for their encoding (4 bits), which gives 36 bits in total. Parts are constructed by a data stream algorithm that processes the values in the array successively as follows. For each value, it checks whether this value can be encoded using the current number of bits. If the outcome of this test is negative, it decides if it is worth to increase the number of bits or to proceeds with a new part. The algorithm in pseudo C is written in Algorithm 5.1.

```
nCurMin = nCurMax = V[0];   //the current minimum and maximum
nCurBits = 1;               //the current number of bits
nProcessed = 1;             //the very first value has been processed

for (i = 1; i < length(V); i++) {
   //get new min and max
   nNewMin = min(nMin, V[i]);
   nNewMax = max(nMax, V[i]);
   nNewBits = number of bits required to store values nNewMin..nNewMax

   //calculate the costs
   nCost1 = 36 + nProcessed*nCurBits + 37;
   nCost2 = 36 + (nProcessed + 1)*nNewBits;
   if (nCost1 < nCost2){
      //start a new part
      store last nProcessed values using nCurBits bits;

      nNewMin = nNewMax = V[i];
      nNewBits = 1;
      nProcessed = 0;
   }

   nCurMin = nNewMin; nCurMax = nNewMax; nCurBits = nNewBits;
   nProcessed++;
}
```

Algorithm 5.1: *The encoding of the array V using variant number of bits.*

5.5 Hilbert Space Filling Curve (BEHEC)

In this method, vertices are processed according to their position on a space filling Hilbert curve [Hil81]. Unlike previously described methods (e.g., VXPATH), we do not encode differences in x and y coordinates (i.e., differences in the Euclidian space) but differences in positions (i.e., differences in a linear space defined by the Hilbert curve). This means that, in

comparison with, e.g., VXPATH, the array *V* has only a half of values but its values might be larger. Figure 5.1 shows the generated arrays *V* and *C* for the image of 8x8 pixels.

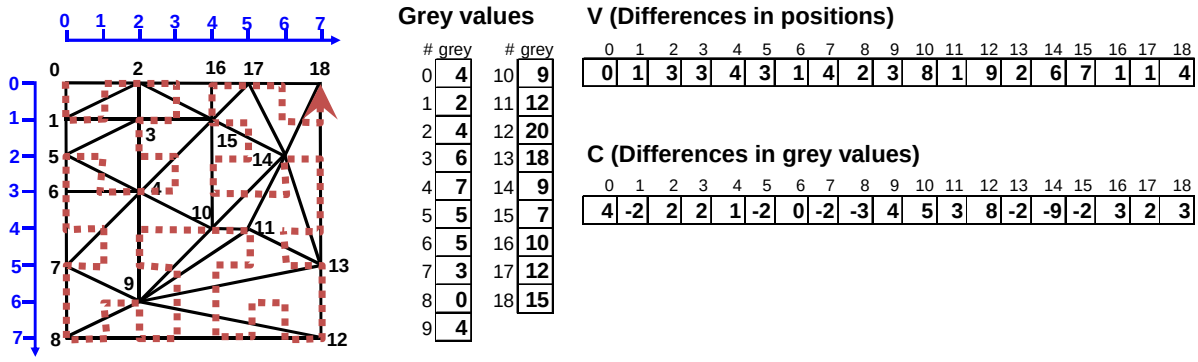


Figure 5.4: *The storing order of vertices and the differences in both positions and grey values. The Hilbert curve is drawn by a dotted poly-line.*

In the next stage of BEHEC, the array *V* is encoded into another array *V'* using variant number of bits (see TRPATH for details) and the final array *V'* is furthermore compressed by the Huffman encoding [Huf52]. The result of the compression, including the constructed Huffman tree (i.e., the dictionary for the encoding), is written into the output file. After that, the array *C*, i.e., the array of differences in grey values is also compressed by the Huffman encoding and the outcome stored into the file.

An advantage of this approach is that it can process the triangulation in $O(N)$ time in the worst case and it is likely to produce very small files. On the other hand, its implementation might be rather complex (especially, the implementation of the Hilbert curve for images of arbitrary sizes). Let us also note that it is quite improbable that additional application of general data compression algorithms would bring a significant change in compression ratio.

5.6 LZ Hilbert Space Filling Curve (LZHEC)

As one can guess from its name, this method also exploits the linearization of vertex space by the Hilbert space-filling curve (see BEHEC). It differs from the just described BEHEC only in the encoding of the arrays *V* and *C*. These arrays are stored in an uncompressed form as follows. Values from the array *V* are processed first. If the value is less than 128, it is stored using 8 bits, i.e., the highest bit is always zero, otherwise, the algorithm transmits one bit set to one followed by bits 8 – 14 of the value and then by its lowest 8 bits, i.e., 16 bits are needed. As it is reasonable to believe that most values will be under the threshold 128, the expected average number of bits per vertex coordinate should not exceed nine.

After that, the array of differences in grey values, the array *C*, is stored using 8 bits per value, i.e., it is processed in the same way as in the VXPATH method. In the last step, the method compresses the output file by some of existing general data compression algorithm. An advantage of this approach roots from its simplicity and efficiency. On the other hand, an external compression utility is required in order to achieve a good compression ratio.

5.7 KORILA

A completely different strategy is exploited in the KORILA method. It is based on the idea presented by Rila et al. [Ril98] to store vertex coordinates as a bitmap that contains 1 at posi-

tions corresponding to the vertices of triangulation and 0 elsewhere. An example of such bitmap is shown in Figure 5.5.

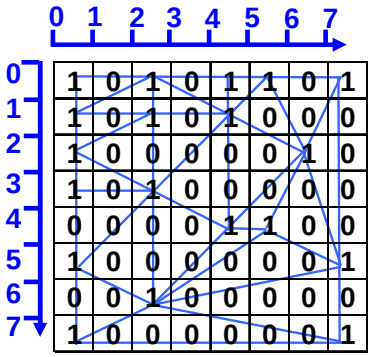


Figure 5.5: *The triangulation and the corresponding bitmap for the 8x8 image.*

After the bitmap is constructed, its bit values are combined in order to get a byte stream, e.g., for our triangulation from Figure 5.5 we would get eight values: 173, 168, 130, 160, 12, 129, 32 and 129. This byte stream is then compressed by the Huffman encoding. Let us note that we also tried to compress the bitmap by the bitwise RLE (Run Length Encoding) [Wic07f] algorithm but it did not bring any improvement. The Huffman has proved to outpace the RLE in all our experiments – see Figure 5.6.

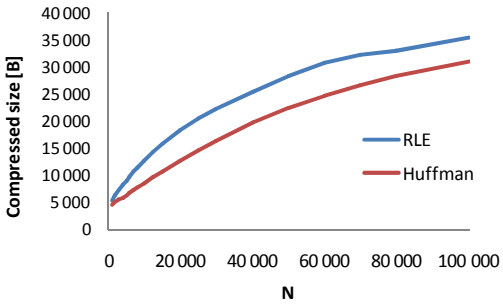


Figure 5.6: *The comparison of average sizes of outputs produced by the bit RLE and the Huffman encoding for triangulations with various numbers of points (N). It was tested on a set of 512x512 grey-scale images.*

Let us suppose to have an image of M pixels that is represented by the triangulation of N vertices (naturally, $N \leq M$). It is obvious that fewer vertices are in the triangulation, the sparser the constructed bitmap is. A different encoding strategy is, therefore, used for small triangulations. A triangulation is considered to be small, if it contains the ratio N/M reaches at most 4% (this threshold was found experimentally). Instead of combining bits into bytes, the bitmap is converted into a one dimensional bit array and this array is split into sequences of zeroes and ones of a predefined maximal length (this value ranges from 16 to 256 according to the ratio N/M). These sequences are, afterwards, used as an alphabet for the Huffman encoding. For the example from Figure 5.5, the alphabet would be 0, 1, 00, 11, 000 and 00000, if the maximal allowed length was 4. In comparison with the byte oriented Huffman encoding used for larger triangulation, more bytes are needed to store the Huffman tree but, on the other hand, the size of compressed data should be lower.

When coordinates are stored, the KORILA method proceeds with storing of grey values. Differences in grey values of vertices ordered just linearly are computed and compressed using the Huffman approach as usual. The topology is not stored at all, i.e., this method is again limited to the encoding of Delaunay triangulations only.

5.8 LZ Image 3D Matrix (LZIM)

Similarly to the approach described by Demaret et al. in [Dem03] (see the Demaret method), the proposed LZIM algorithm starts with the construction of 3D binary matrix that contains $W \times H \times Q$ cells, where W and H are the width and height of the input image and Q denotes the number of supported grey values. As we do not quantize grey values stored in vertices of the triangulation (unlike Demaret et al.), Q is always 256. A cell at the position $[i, j, k]$ holds one if and only if there is a vertex having the coordinates $[i, j]$ and the grey value equals to k, otherwise zero value is stored in this cell.

In the next stage of this method, the 3D matrix, which is very sparse, is stored into the output file using one byte per value and this file is afterwards compressed by some general data compression algorithm. An advantage of this approach is its simplicity and efficiency. On the contrary, the uncompressed output file is huge (e.g., for a quite small image of 512x512 pixels, 64 MB are consumed).

5.9 Mueller (MUEKD)

Starting with the construction of 3D binary matrix (see LZIM), here aka as the box, the MUEKD algorithm continues with a recursive subdivision of this box by axis aligned cuts. In each step of the recursion, the box is split into two smaller boxes having as equal number of cells storing the value one as possible. In an ideal case, both boxes contain the same number of ones. Let us note that we always construct a cut such that it subdivides the longest side of the box. The recursion stops when cells in the box are uniform, i.e., they hold the same value.

In order to speed up the processing, we exploit summed-area table of the matrix [Mue97]. Summed-area table is a three-dimensional array such that the value of its cell at the position $[i, j, k]$ is equal to the sum of the values of the cells in the original matrix at positions [0 up to i, 0 up to j, 0 up to k]. Figure 5.7 shows an example of binary matrix in E^2 and its corresponding summed-area table. Let us note that the summed-area table can be efficiently found in $O(R)$, where R is the total number of cells.

0	0	0	0	0	0	0	0
1	0	1	0	1	0	0	0
1	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0
0	0	0	0	1	1	0	0
1	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	1

a) bitmap

0	0	0	0	0	0	0	0
1	1	2	2	3	3	3	3
2	2	3	3	4	4	5	5
3	3	5	5	6	6	7	7
3	3	5	5	7	8	9	9
4	4	6	6	8	9	10	11
4	4	7	7	9	10	11	12
5	5	8	8	10	11	12	14

b) summed-area table

Figure 5.7: *The construction of summed-area table in E^2.*

Let us explain the process of subdivision using the summed area table by an example in E^2 – see Figure 5.8. In the first step, we divide the summed-area table into two parts x-coordinate. In an ideal case, both parts would contain seven ones. It is, however, impossible to find an exact position of cut to achieve this ideal case. All that can be done is to find a cut that mini-

mize the deviation from the ideal case as follows. First, the proper position of value 7 in the one-dimensional array 5, 5, up to 14 (i.e., the last row in the table) is found. As this array is ordered, we can use modified binary-search algorithm for this purpose. The value somewhere lies between values 5 and 8. It is necessary to decide whether the value 8 will belong to the first part or the second one. The former option introduces the error 7-5=2 and the latter the error 8-7=1. Therefore, the cut between the values 8 and 10 is created and the summed-table area updated as Figure 5.8b shows.

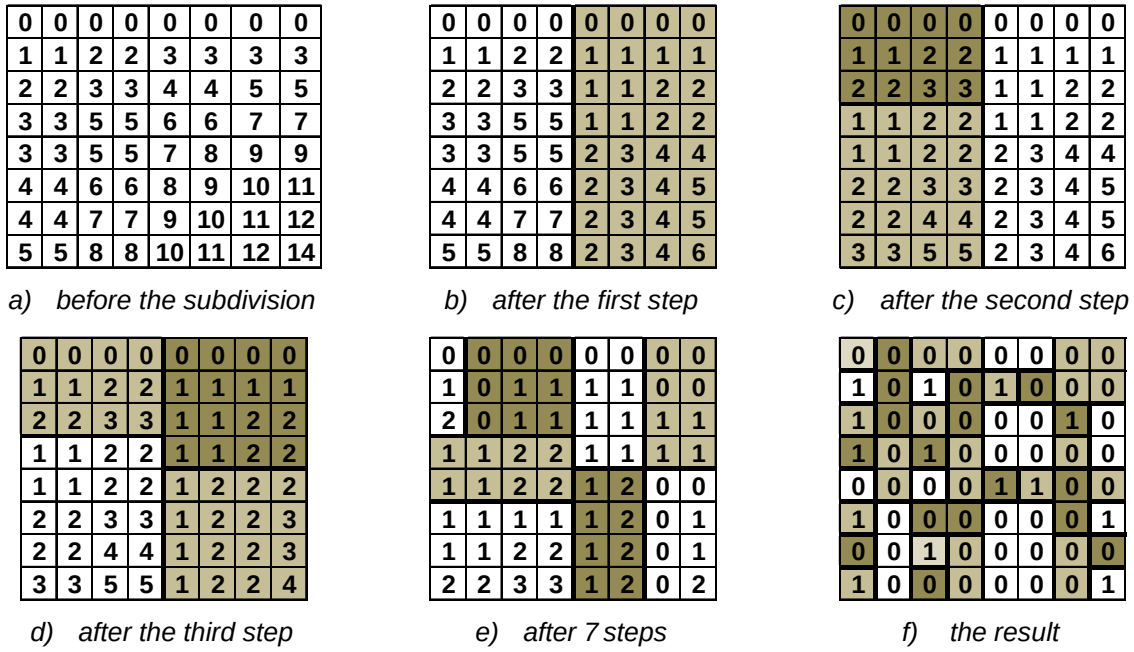


Figure 5.8: *The recursive subdivision of the summed-area table in E^2.*

In the second step, we divide the first part in y-coordinate (it is the longer one). The position of the value 4 in vector 0, 2, etc. (i.e., the last column in the area) is found. It is between values 3 and 5. The table is subdivided and its values are updated (see Figure 5.8c). The algorithm continues until entirely zeroes or ones areas are achieved as shown in Figure 5.8f.

The history of subdivision process is kept in a binary tree. Its inner nodes represent areas that were subdivided and each of these nodes can be encoded by one zero bit followed by the relative position of cut used to subdivide the corresponding area. For the position, $\lfloor \log_2(s-1) \rfloor$ bits where s is the length of divided side is used, i.e., at most 3 bits are consumed per cut for our example from Figure 5.8. Leaves representing zeroes areas can be encoded using two bits long code 10 and leaves representing ones areas can be encoded using code 11.

Let us, however, discuss the case when we have an area where all cells are zeroes but those on the diagonal – see Figure 5.9. We need 18 cuts whose encoding would consume 50 bits and the encoding of constructed 19 areas would take other 38 bits, which gives that 88 bits would be required. If we, instead of subdividing the area, stored (linear) positions of cells with the value one, i.e., we store numbers 0, 8, 16, 24, 32, 40 and 48, using 6 bits per one value (as there is 49 cells, 6 bits are sufficient), only 42 bits would be consumed. As an image edge is typically represented by more vertices than the remaining parts of image, the case we have just discussed is very likely not a singular one but it may appear in real data quite often.

a) the bitmap *b) the result of subdivision*

Figure 5.9: *The bitmap (in E^2) and its subdivision.*

Therefore, we decided to use three bits long code 110 for leaves representing zeroes areas and four bits long code 1111 for leaves representing areas with ones cells and to modify the approach as follows. Starting from leaves, the methods checks for every inner node whether it is worth to split the area or to store positions of its zeroes or ones. If positions are to be stored, four bits long code 1110 for zeroes and two bits 10 for ones is written into the output stream first followed by the number of values that follows. Figure 5.10 brings this "merging step" for an example shown in Figure 5.8. As it can be seen, the minimal storage requirements are achieved if the initial area is split vertically (bits 0 110 are transmitted), its left part is furthermore split horizontally (bits 0 100 transmitted) and then positions of ones in areas denoted in the figure as 18, 32 and 44 are stored.

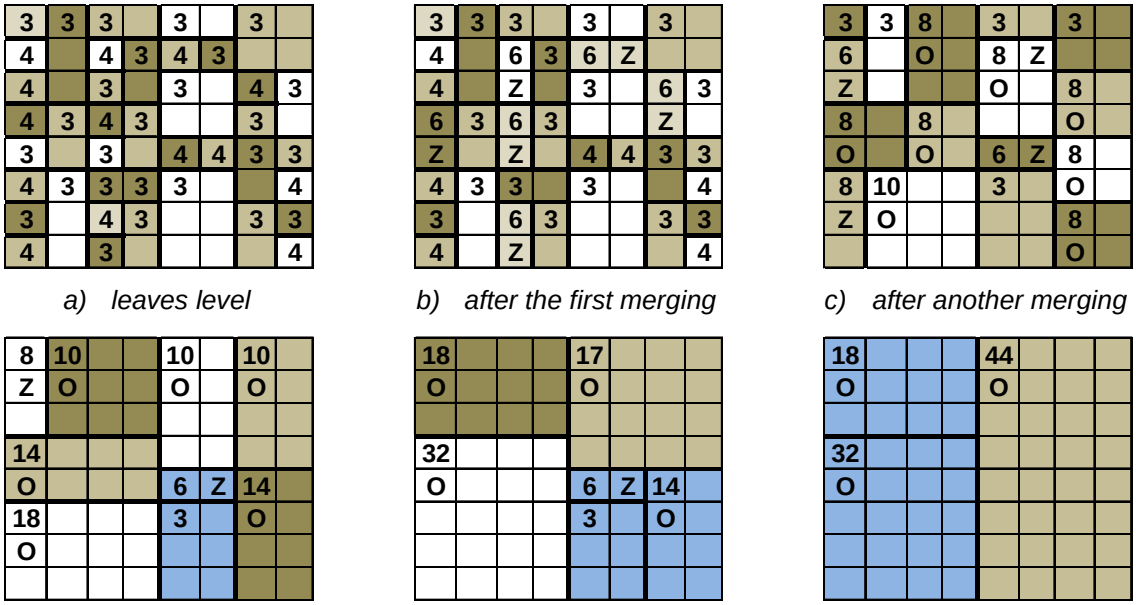


Figure 5.10: *The evaluation of encoding costs. Symbols O and Z denote areas for whose positions of cells holding the value 1 (O) or 0(Z) are to be stored.*

Although this method may seem to be complex, its implementation is quite simple. Its great advantage is that it promises low storage costs. On the other hand, despite the use of summed-area table, it may be rather slow and what is more important it consumes a lot of memory as 4 bytes per one cell of summed-area table are needed, i.e., to process an image of 1024x1024 pixels one would need 1GB of memory.

5.10 Demaret

Besides the methods we have described in previous section, we also experimented with several existing methods. Demaret et al. [Dem03] propose an approach suitable for the compression of Delaunay triangulations representing images of $2^k \times 2^k$ pixels. It stores the number of vertices using $2 \cdot k$ bits into the output file first and then it continues with the quantization of grey values held in vertices followed by the construction of 3D matrix (see MUEKD), here also known as the box. Let us note that we cease the quantization in our implementation in order to be able to compare this method with our methods.

Afterwards, the box is recursively divided. In every step of recursion, the box is successively split by three axis-aligned cuts chosen in the middle of every side of the box into eight smaller boxes – see Figure 5.11. The algorithm stores numbers of vertices in areas L, LT, LTN, LBN, RT, RTN and RBN using only as many bits as necessary, e.g., in the very first step of recursion, the value for the area L is encoded using k bits, values for LT an RT use k-1 bits and the remaining values are stored on k-2 bits. Let us note that if any area is empty, i.e., it contains no vertex in its interior, it is not divided any more. For an instance if the area R was empty, values for RT, RTN and RBN areas would not be neither computed nor stored.

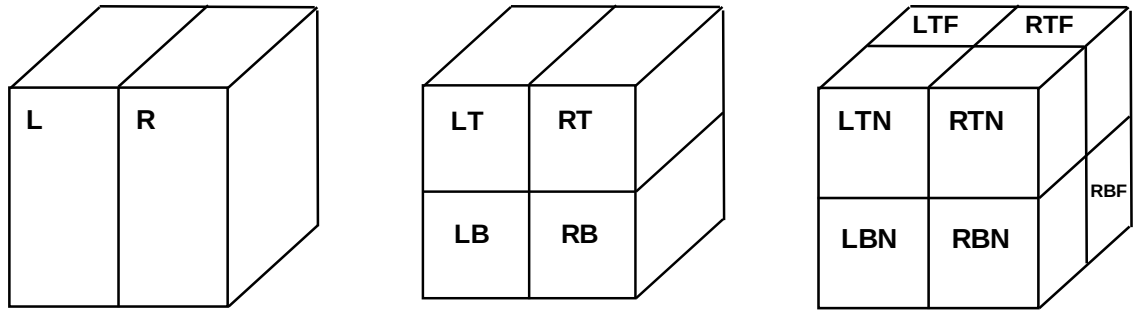


Figure 5.11: *The box subdivision.*

The recursion stops when either the box to be subdivided is empty or it is formed by 2×2×1 cells. For the latter case, the algorithm stores a short binary code according to the configuration of zeroes and ones in cells into the output file. Figure 5.12 brings these codes.

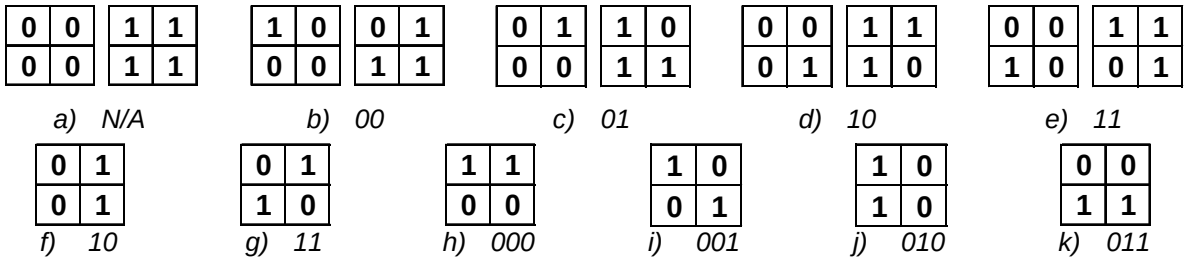


Figure 5.12: *Binary codes for various configurations.*

A drawback of this approach is that it consumes a lot of memory (because of 3D matrix). Its generalization for any image size is possible although not easy to be implemented. On the other hand, according to the published results, this encoding strategy can achieve compression ratio comparable even with JPEG 2000.

5.11 Demaret06

Another method proposed by Demaret et al. [Dem06] starts with the construction of bitmap representing coordinates of vertices (see the KORILA method). The bitmap is an initial area that is recursively split into two smaller rectangular areas of equal size, if the area height is larger than its width, by a horizontal cut, otherwise, by a vertical cut. The splitting terminates at areas that are either empty, i.e., not containing any vertex or atomic, i.e., they are of one pixel size. Whenever an area is subdivided, the number of vertices lying in the first part (i.e., in top or left area) is written into an array of 32-bits integers. An example of bitmap subdivision is shown in Figure 5.13.

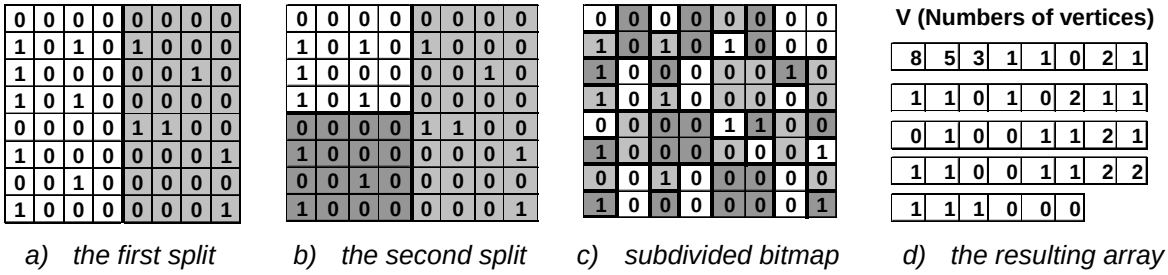


Figure 5.13: *The bitmap subdivision.*

The constructed array of numbers is compressed by the Huffman encoding and the result is stored into the output file. After that, grey values held in vertices are also compressed by the Huffman encoding and stored. Let us note that authors perform the quantization of grey values in prior to encoding but we skip this step in our implementation so it can be compared with the other approaches.

The method is simple to be understood and implemented. According to the results presented by authors, it also promises a good compression ratio.

5.12 Edgebreaker

Edgebreaker proposed by Rosignac [Ros99] is probably the most often used algorithm for a compression of an arbitrary triangulation. Edgebreaker visits triangles in a spiralling order and generates a string of symbols from the set {C, L, E, R, S}. This string describes the topology, i.e., it indicates how the mesh can be rebuilt. Using the Huffman compressor, it can be efficiently encoded so that two bits per triangle are guaranteed. The geometry, i.e., the coordinates and grey levels of vertices are encoded as follows. When a triangle, say p_a, p_b, p_c, is visited and the far vertex p of its adjacent triangle has not been processed yet, a prediction q is computed using the parallelogram predictor and the algorithm stores the differences between this predictor and the vertex p into arrays V and C – see Figure 5.14.

Both arrays of differences are stored into the output file using a fixed number of bits per vertex (see VXPATH). This number depends on how accurate the predictions are. Theoretically, if the predictor were able to give always an accurate prediction (i.e., q and p are the same), it would be possible to avoid the storing of geometry. In practice, however, this case does not exist.

An advantage of this algorithm is that it can compress any triangulation, not only the Delaunay triangulation, and, if the parallelogram predictor is used, it runs incredibly fast. On the other hand, its implementation for triangulations with boundaries is not as simple as for

closed meshes. Another drawback is that the topology has to be stored even for the Delaunay triangulation because it is needed for decoding of coordinates and grey values.

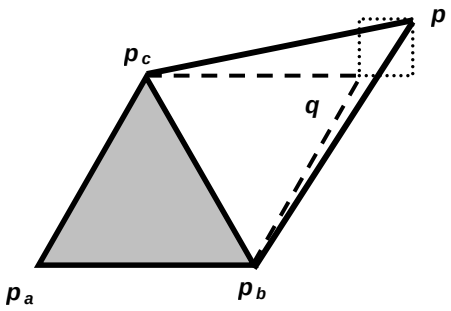


Figure 5.14: *The prediction q of the vertex p.*

5.13 Coddyac

Another approach suitable for the compression of an arbitrary triangulation, with the code-name Coddyac, is described in [Vas07]. It is a lossy compression originally proposed for encoding of dynamic meshes, i.e., for triangulations whose vertices may change their location in time. Authors clearly demonstrate that their approach outperforms EdgeBreaker and, therefore, a better compression ration might be expected (in trade of quality).

Similarly to Edgebreaker, the algorithm stores differences between vertices and their predictions. Instead of the parallelogram predictor, a predictor based on PCA (Principal Component Analysis) [Wic07e] is exploited. Let us note that as the PCA is very time consuming, the compression of even small triangulation (having a couple of thousands vertices) might take several minutes on commodity hardware.

6 Experiments with Triangulation Encoding

In the previous section, we described various methods for the compression of triangulations representing grey-scale images. Most of them do not store the connectivity of vertices and, therefore, they are suitable for the Delaunay triangulation only because this kind of triangulations can be recomputed from vertices in the reconstruction process. All these methods were tested on Delaunay triangulations of grey-scale images from the tested set that were produced by the BRUTE decimation technique (see Section 3, 4).This section brings results of our experiments.

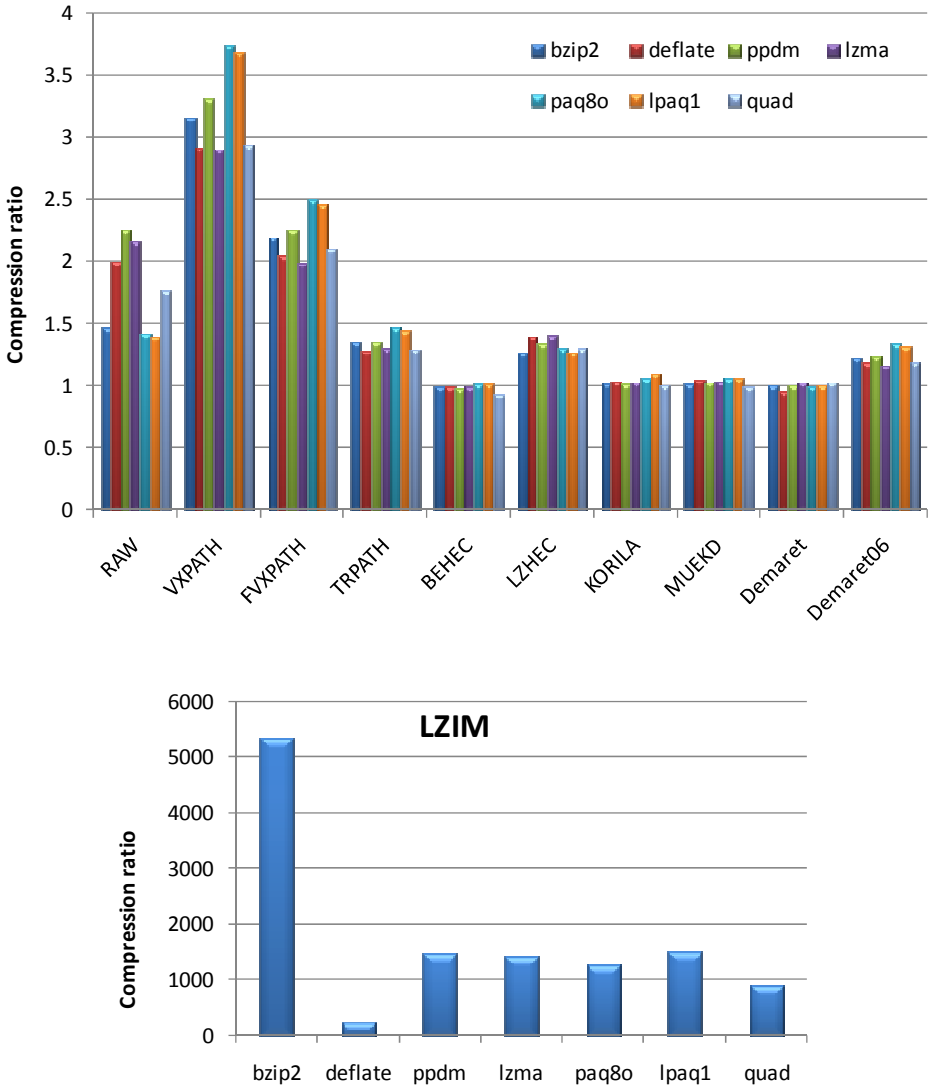


Figure 6.1: *Average compression ratios achieved by data compression utilities for outputs produced by various methods of the Delaunay triangulation encoding.*

6.1 compares average compression ratios that were achieved by various general data compression algorithms for Delaunay triangulations encoded by methods described in the previous section. As it can be seen, there is no significant difference in these compression algorithms for triangulations encoded by methods that use a variable number of bits for the encoding, i.e.,

TRPATH, BEHEC, LZHEC, KORILA, MUEKD, Demaret and Demaret06. For the remaining methods, there is no universal compression algorithm; a good choice could be bzip2 (especially, for LZIM encoded triangulations) or paq8o. If we take all achieved compression ratios into an account, the winning algorithm is paq8o (with the score 92.2%) followed by lpaq1 (score 91.6%), bzip2 (90.5%) and then lzma (85.7%), ppdm (84.6%), deflate (81.4%) and quad (80.4%). Nevertheless, both paq8o and lpaq1 algorithms are, especially for larger files, time consuming. For an instance, a file produced by LZIM was typically compressed by paq8o in 18 – 25 minutes, while bzip2 needed just a few seconds. Therefore, we consider bzip2 to be the best choice.

As it can be also seen, while outputs of methods that store data without any use of sophistic encoding techniques (e.g., the Huffman encoding, RLE) are well compressible, outputs of BEHEC, KORILA, MUEKD and Demaret methods are not compressible at all. Surprisingly, the Demaret06 method, although it exploits the Huffman encoding, produces files that can be compressed by any of considered data compression algorithms. The explanation is simple, if we recall the character of input data processed by the Huffman encoding (see Figure 5.13d). The most frequent value in the input data is zero and, therefore, it will be likely encoded using one zero bit only. As there are long sequences of zeroes in the input data, it is very likely to have sequences of zero bytes also in the output files that can be, indeed, well compressed in the post-processing by some general data compression algorithm.

Last thing visible in the graph is that by storing differences in coordinates instead of storing pure coordinates we get results that are better compressible – compare the compression ratio for RAW and VXPATH methods. Without any surprise, data encoded using a fixed number of bits per value are also better compressible than data encoded using a variable number of bits – see VXPATH, FVXPATH vs. TRPATH.

A comparison of methods for the encoding of Delaunay triangulations (i.e., they do not store the connectivity of vertices) is given in Figure 6.3 – Figure 6.7. Unsurprisingly, the RAW method produces much larger files (even if they are compressed by bzip2) than others. Poor results are also achieved by the TRPATH method. The reason roots probably from larger differences (see Section 5.4), whose storing require a lot of bits.

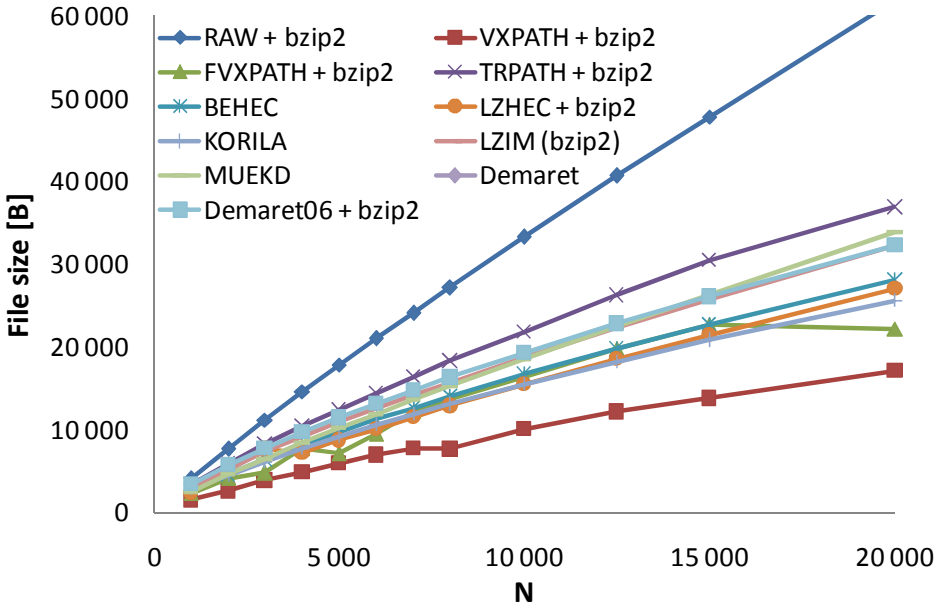


Figure 6.2: *The comparison of file sizes achieved by various encoding methods for Delaunay triangulations of various numbers of vertices (N) that represent the Maran image (400x300).*

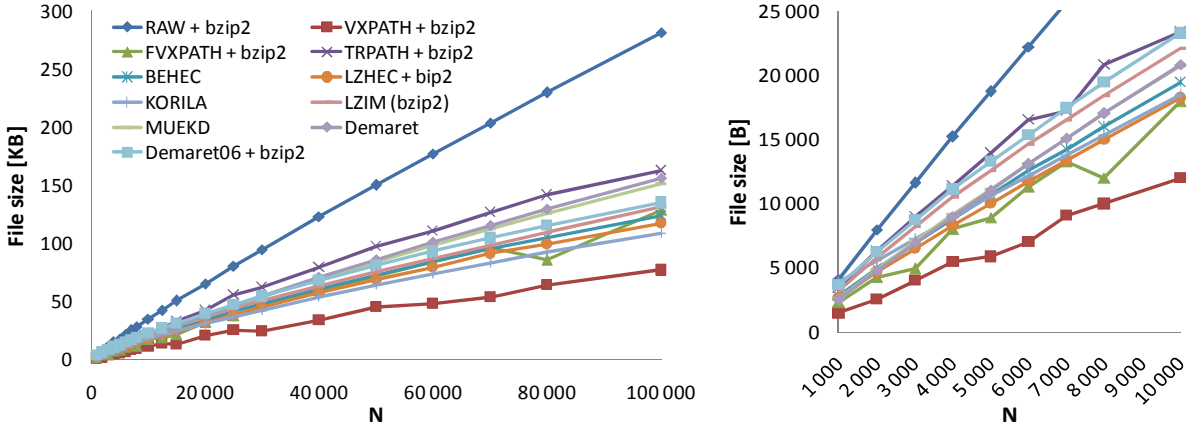


Figure 6.3: *The comparison of file sizes achieved by various encoding methods for the image of boats (512x512) represented by Delaunay triangulations of various numbers of vertices (N).*

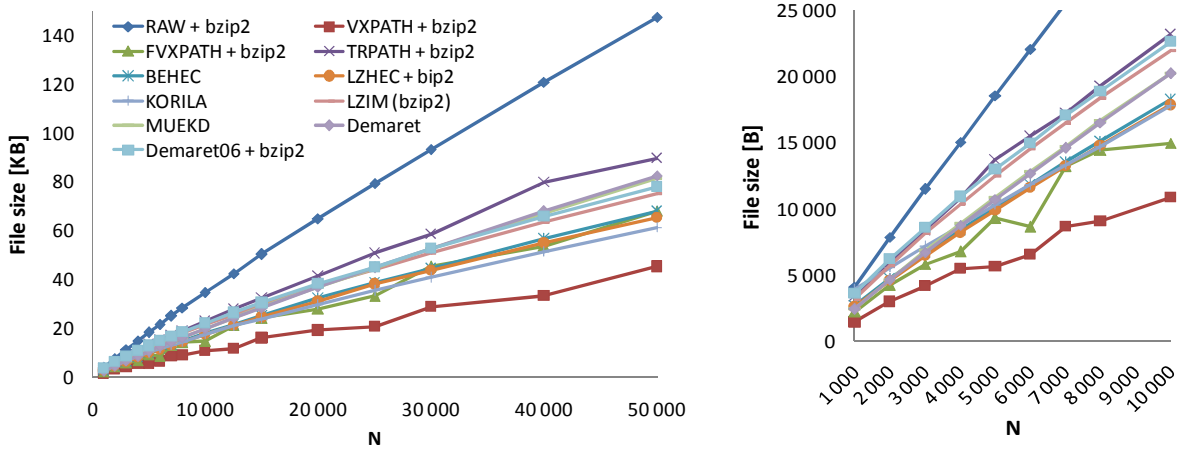


Figure 6.4: *The comparison of file sizes achieved by various encoding methods for the image of fruits (512x512) represented by Delaunay triangulations of various numbers of vertices (N).*

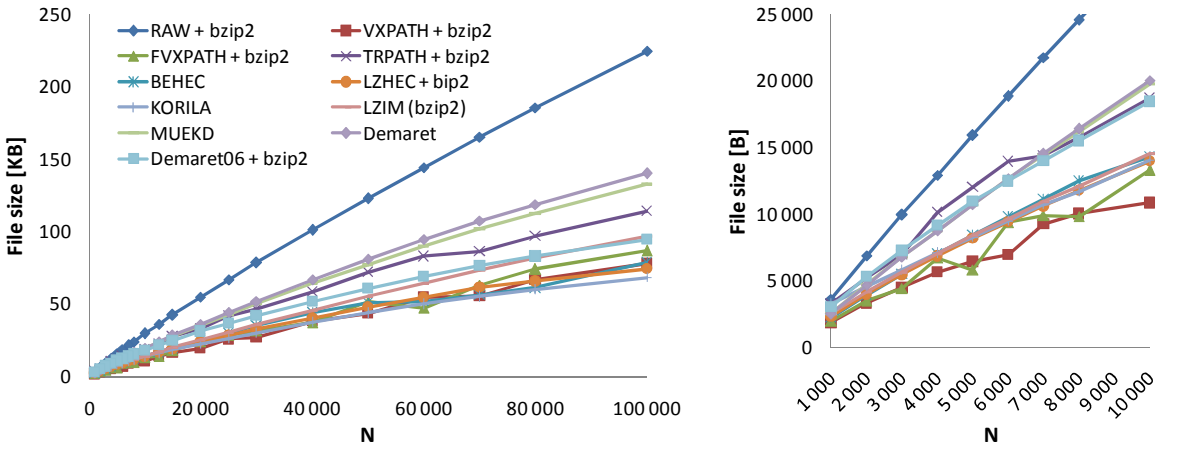


Figure 6.5: *The comparison of file sizes achieved by various encoding methods for the Lena image (512x512) represented by Delaunay triangulations of various numbers of vertices (N).*

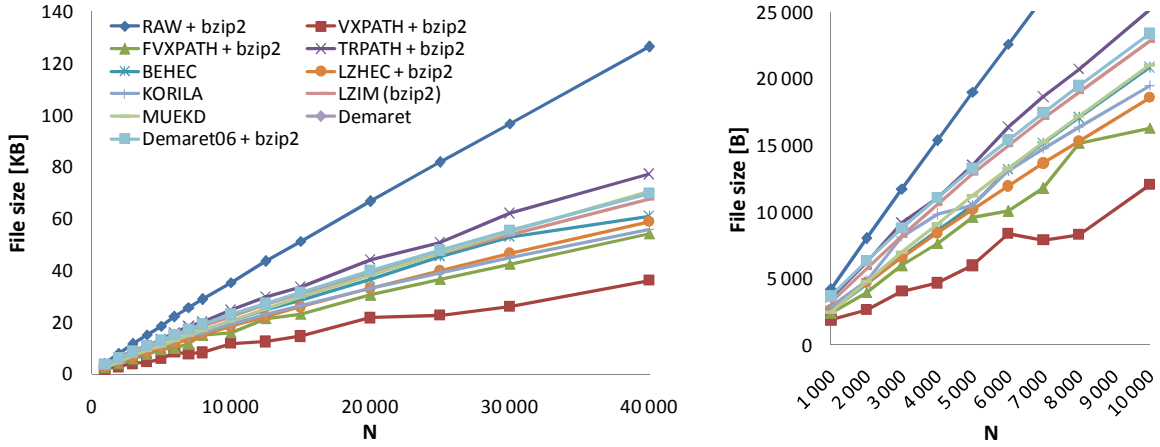


Figure 6.6: *The comparison of file sizes achieved by various encoding methods for the image of monarch (768x512) represented by Delaunay triangulations of various numbers of vertices (N).*

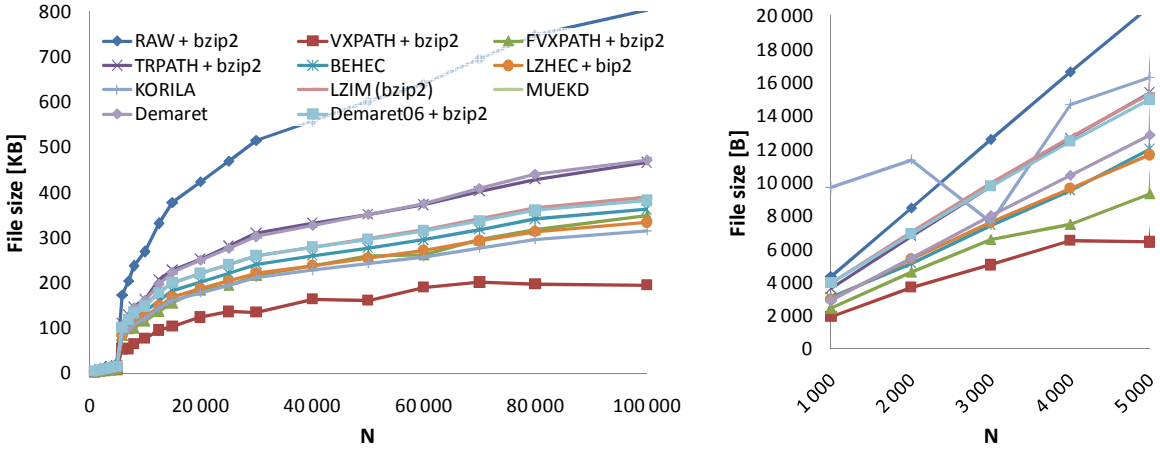


Figure 6.7: *The comparison of file sizes achieved by various encoding methods for the image of pirate (1024x1024) represented by Delaunay triangulations of various numbers of vertices (N).*

As it can be seen, there is no significant difference between Demaret and MUEKD methods. For larger triangulations (with more than 10 000 vertices), these methods are slightly outpaced by Demaret06. The smallest files were achieved by the VXPATH encoding combined with additional compression by bzip2. This technique is typically followed by its faster version, the FVXPATH. All other methods produce files with sizes between sizes achieved by VXPATH and TRPATH.

We also tested Edgebreaker and Coddyac methods that are suitable for encoding of arbitrary triangulations. Table 6.1 compares the results obtained by Edgebreaker for triangulations with 50 000 vertices. While VXPATH and TRPATH methods achieved a similar compression ratio for all three triangulations, Edgebreaker shows a different behaviour. The explanation is quite simple. The regularity of constructed triangulations is significantly influenced by the amount of sharp edges and the richness of objects as well as by the range of grey values in the image. For an instance, the Lena image contains 28 grey values and a few objects only, whilst the image of fruits consists of many objects of 256 grey values. If the triangulation is very irregu-

lar, as it can be seen in Figure 6.8, the predictor used in Edgebreaker often produces highly inaccurate predictions, which leads to lower compression ratio.

Image	PSNR	Edgebreaker	VXPATH	TRPATH
Lena	36.29	152 624	175 012	138 450
Fruits	39.95	171 464	162 512	156 106
Boat	35.30	273 648	175 012	144 288

Table 6.1: *The comparison of sizes of output files produced by various compression schemes for triangulations with 50 000 vertices of three popular 512x512 grey-scale images.*

While the VXPATH method in its basic form needs typically less than 3.5 bytes, i.e., 28 bits, to encode one vertex (both coordinates and grey values), Edgebreaker requires 2.5 up to 4.5 bytes per vertex and at most 2 bits per triangle. If we consider that there are twice as many triangles as vertices in a triangulation, it gives us that the Edgebreaker method needs 24 up to 40 bits to encode one vertex. In an average case, VXPATH takes about four bits per vertex less than Edgebreaker. It means that if we have an image represented by the Delaunay triangulation of N vertices, Edgebreaker is worth using only in such a case that one can represent this image (in the same quality) by an arbitrary triangulation of $2/3 \cdot N$ vertices only. In our opinion, this is a very difficult (if not even impossible) task.

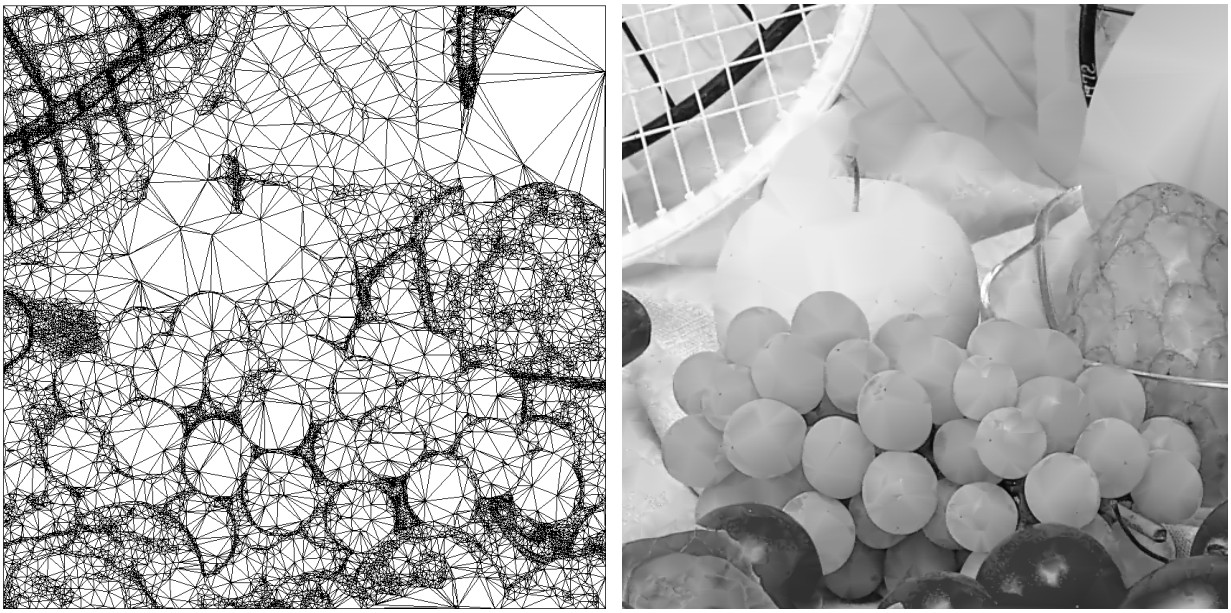


Figure 6.8: *The Delaunay triangulation with 15 000 vertices and the corresponding reconstructed image (PSNR = 33.90).*

When we tried to zip output files in order to achieve a better compression ratio, Edgebreaker has never outperformed the VXPATH method – see Table 6.2.

Image	PSNR	Edgebreaker	VXPATH	TRPATH
Lena	36.29	66 566	52 470	90 602
Fruits	39.95	79 620	57 135	100 998
Boat	35.30	110 419	53 677	115 788

Table 6.2: *The comparison of sizes of zipped output files produced by various compression schemes for triangulations with 50 000 vertices of three popular 512x512 grey-scale images.*

The Coddyac method has proven to be useless. This method required almost half an hour to encode a triangulation of 3991 vertices, while the remaining methods do this in a couple of seconds or at most in several minutes. It, moreover, produced files with sizes that range from 12 to 38 KB according to the chosen encoding quality (let us remind reader that Coddyac is a lossy compression technique). It leads to an obvious conclusion that it makes no sense to employ neither Edgebreaker nor Coddyac for the encoding of triangulations.

We compared the achieved results also with JPEG and JPEG2000. As it can be seen from Table 6.3, both compression techniques outpace even the best VXPATH method. Let us note that both Demaret and Demaret06 methods, if the quantisation of grey values were used, should have been, according to authors claims, competitive to wavelet encoding exploited in JPEG2000. Without the quantisation, however, both methods are beaten by LZHEC, FVXPATH and even by VXPATH. Thus it seems reasonable to believe that if grey values were quantized, VXPATH should outperform JPEG2000.

		VXPATH + bzip2			JPEG			JPEG2000	
Lena	Size	44 310	19 866	3 329	38 467	16 159	3 147	25 538	11 948
	PSNR	36.26	33.41	24.68	36.22	33.46	24.73	36.34	33.37
Fruits	Size	46 345	19 521	3 002	45 584	21 367	2 677	33 081	15 755
	PSNR	39.95	35.27	24.52	39.31	35.28	24.58	39.97	35.31
Boat	Size	46 724	21 404	3 996	39 357	16 295	3 024	26 378	12 325
	PSNR	35.30	31.00	23.35	35.36	31.00	23.03	35.26	30.95

Table 6.3: *Sizes of zipped output files produced by the VXPATH method for different triangulations of popular 512x512 grey-scale images in comparison with sizes produced by JPEG and JPEG2000.*

From results presented in this section, it is quite clear that the compression of computed triangulation is a very important issue. Even a small change of an existing method could dramatically change its typical compression ratio. Performed experiments show that it would be probably fruitless to construct an arbitrary triangulation instead of the Delaunay one because, although it would contain fewer vertices, it cannot be stored using fewer bytes. Apparently, the storing of differences using a fixed number of bits followed by a general data compression is better than storing them using a variable number of bits. It seems furthermore that the quantization of grey values plays an important role. In our future work, techniques identified during our experiments to be worthy should be combined together in a new method.

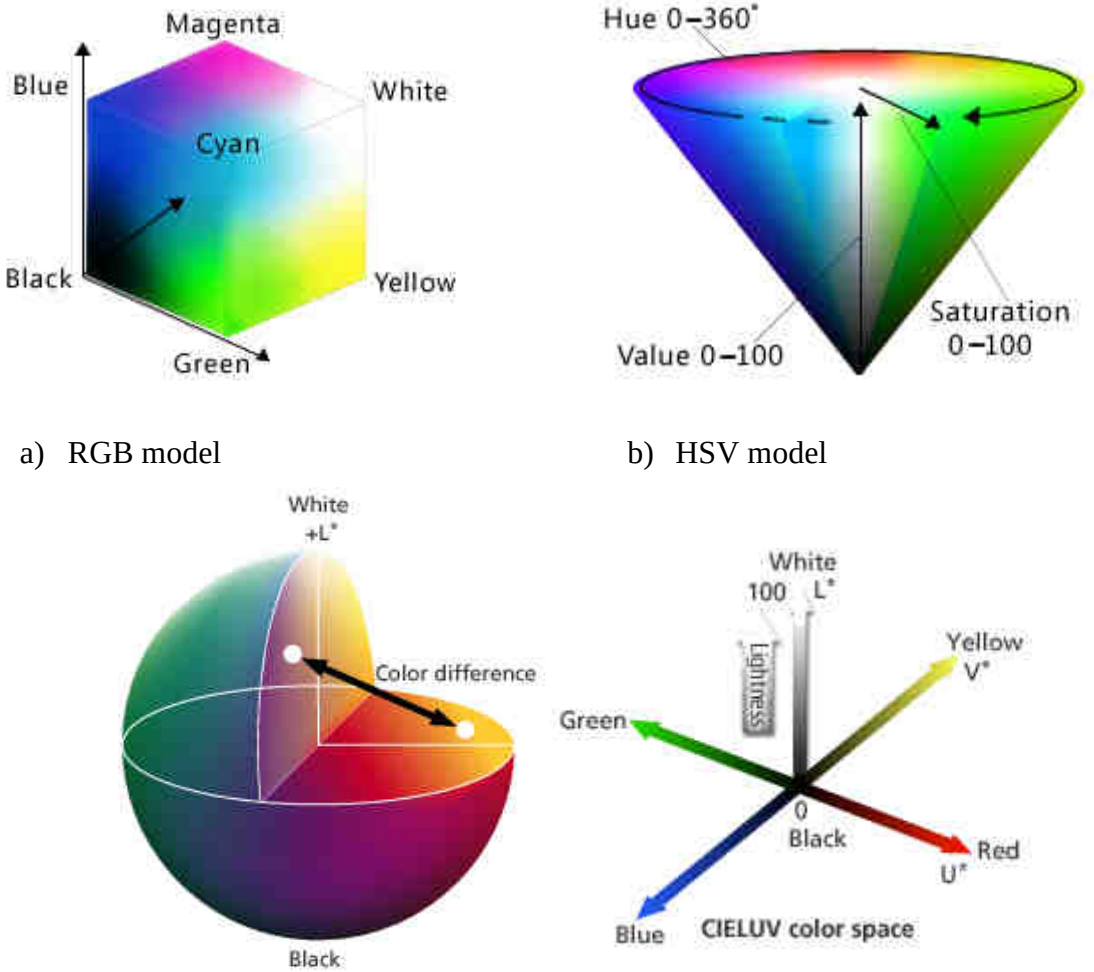
7 Extension for Colour Images

A pixel in a colour image is represented by a vector of three (or four) components whose meaning depends on the colour model used for the representation. Therefore, a straightforward extension of the proposed alternative representation of images by the Delaunay triangulation (see sections above) is to deal with each colour component independently and construct three (or four) component triangulations that are afterwards encoded giving three (four) output files. In our research, we experimented with images represented using RGB, HSV, L*u*v and YCbCr colour models.

7.1 Colour Space Systems

The RGB colour model (see Figure 7.1a) is an additive colour model in which red, green, and blue light are added together in various ways to reproduce a broad array of colours. The main purpose of this model is for the display of digital images in electronic systems (such as televisions and computers), which renders this model to be the most commonly used one.

The HSV colour model better corresponds to the human perception of colours (and their blending) as the colour is defined by hue, saturation and lightness value – see Figure 7.1b. This model is widely used in applications for user manipulation with digital photographs.



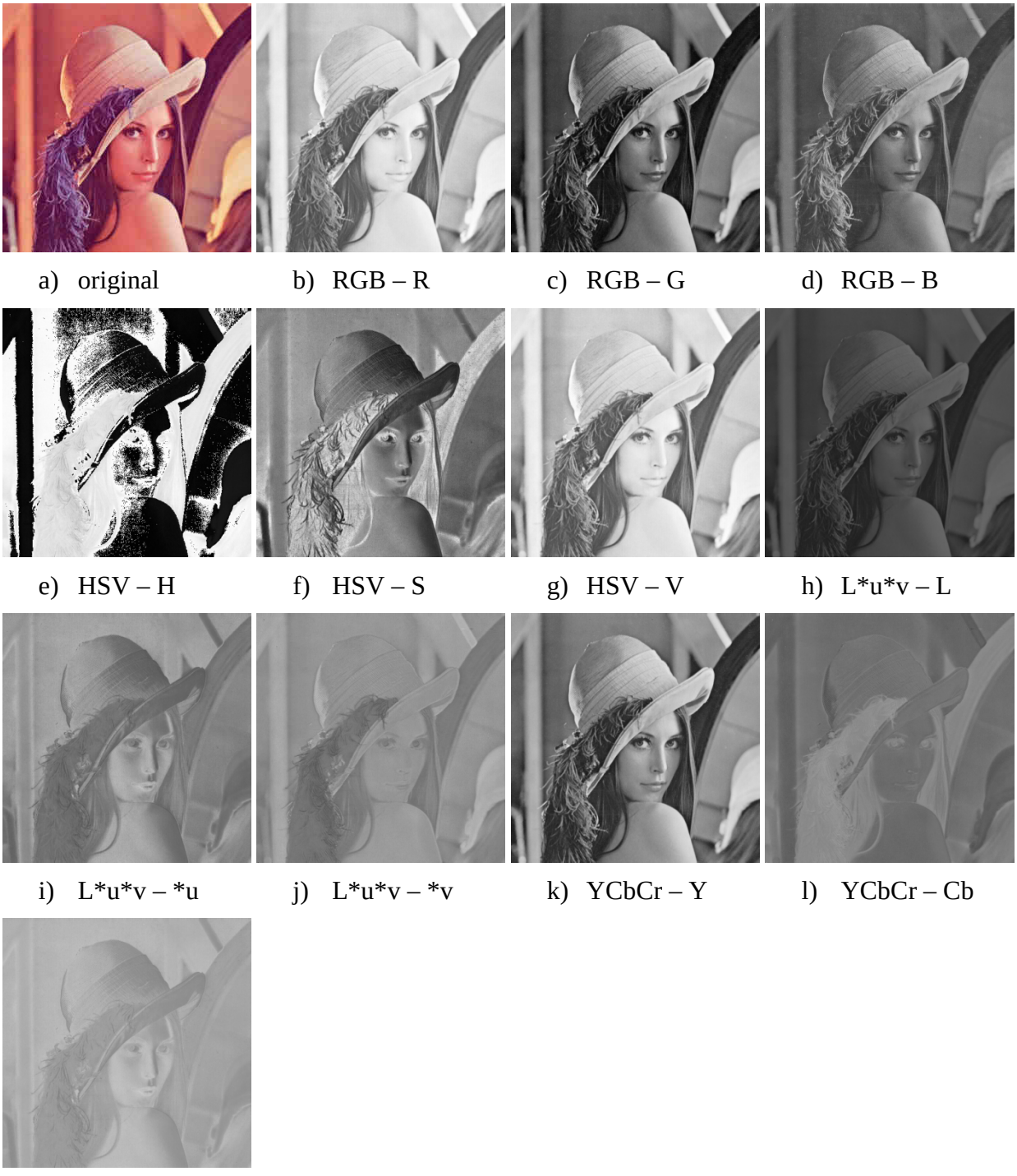
a) RGB model

b) HSV model

c) CIE L*u*v model

Figure 7.1: *Various colour models [MSDN09, Nik09].*

The CIE L*u*v* colour model, also known as the CIELUV colour model, is a device independent colour space system based on the idea that red and green and blue and yellow as well are distant colours. Hence, it is possible to introduce values describing the position of colour between red and green (component *u) and between blue and yellow (component *v). The component L then defines lightness – see Figure 7.1c. The L*u*v model is extensively used for applications such as computer graphics which deal with coloured lights.



a) original b) RGB – R c) RGB – G d) RGB – B

e) HSV – H f) HSV – S g) HSV – V h) L*u*v – L

i) L*u*v – *u j) L*u*v – *v k) YCbCr – Y l) YCbCr – Cb

m) YCbCr – Cr

Figure 7.2: *Separated colour components of Lena colour image in various colour systems.*

In the YCbCr model, Y is the luminance component and Cb and Cr are the blue-difference and red-difference chromatic components, respectively. Originally it was developed for SECAM TV system and later adopted in various video and image encoding methods such as JPEG or MPEG. Its main advantage is that due to the separation of luminance, it is possible to transmit the luminance (Y component), which is more important for human perception, in a high resolution and chromatic components (Cb and Cr), not so important for human perception, in a lower resolution.

A colour represented in one colour space model can be easily transformed to be represented in another colour space model. This transformation, however, is not always without a loss of information (especially in case of device dependent colour systems). Colour components of Lena image represented using various colour models are shown in Figure 7.2. As it can be seen, while the range of values of RGB and HSV components is large, i.e., it is reasonable to expect lot of vertices in component triangulations in order to achieve a good quality, the range of values of components *u, *v, Cb and Cr is quite low, i.e., these components can be likely represented by triangulations with fewer vertices than in case of L and Y components.

7.2 Separate Triangulations

Figure 7.3 presents the dependency of the quality of the component triangulation (measured in PSNR) on the number of vertices for Lena colour image. Let us note that similar results are, indeed, achieved also for other tested colour images. Unsurprisingly, the degradation in the quality might be significantly faster in one component than in the other one (for instance, compare L with *u or *v components). The performed experiments confirmed our hypothesis that Cb and Cr components need fewer vertices than the component Y to be represented in the same quality. This is especially true for airplane, baboon, fruits, tulips and other images whose results are not presented in this report. Unlike our expectation, however, experiments show that the component L (in L*u*v colour model) is less important than the remaining components, i.e., it needs fewer vertices.

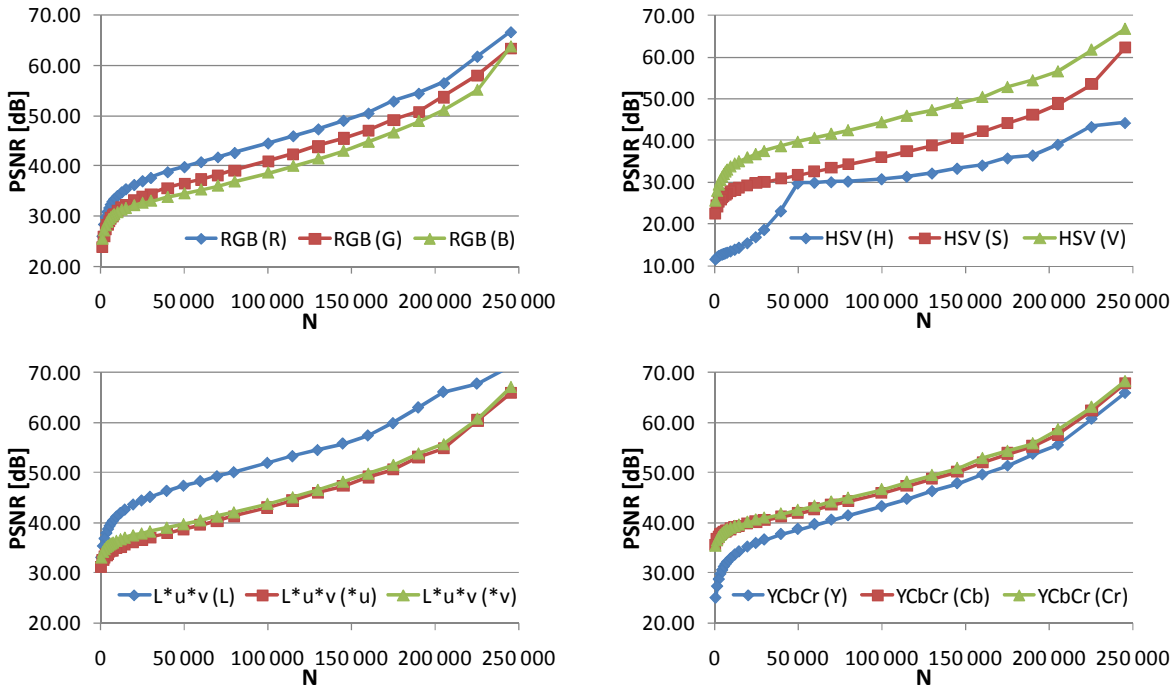


Figure 7.3: *The dependency of quality (measured in PSNR) of images reconstructed from Delaunay triangulations on the number of vertices in component triangulations for colour 512x512 Lena image.*

Component triangulations with the same (or at least similar) quality are grouped together to form the final geometric representation the colour image. Figure 7.4 brings a comparison of the achieved quality for various colour models in the dependence on the overall amount of vertices in component triangulations. While the RGB model clearly outpaces the other tested models when larger triangulations are considered, it achieves almost the same results as the other models (except for the HSV model) for smaller triangulations. As smaller triangulations are typically required to get a good compression ratio (see Section 6), we could conclude that any colour space except for the HSV is suitable for the proposed geometric representation.

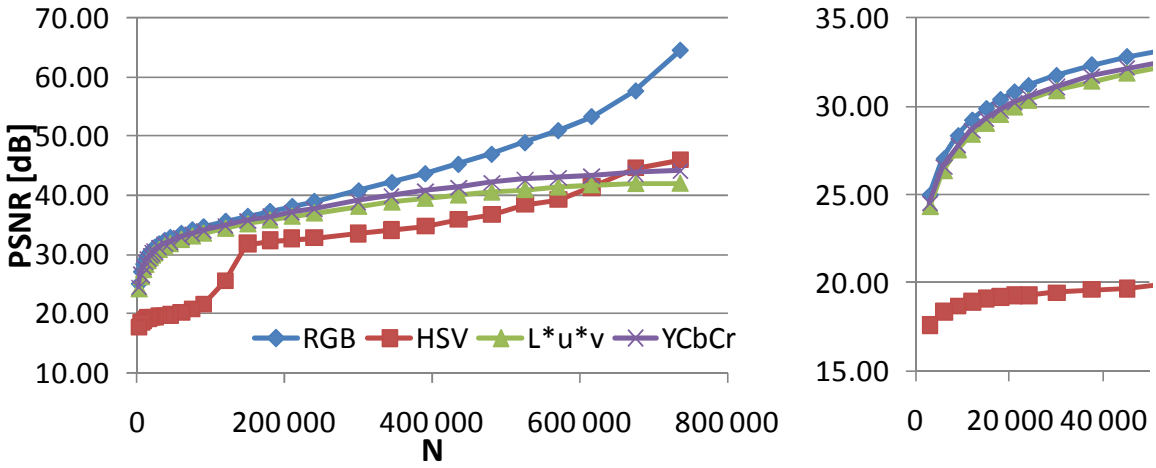


Figure 7.4: *The dependency of quality (measured in PSNR) of images reconstructed from Delaunay triangulations on the overall number of vertices (in all component triangulations) for colour 512x512 Lena image.*

A relationship between the size of final triangulation (composed of three independent component triangulations) and the size of output file (every component triangulation is encoded independently producing three output files whose sizes are summed to get the final size) is depicted in Figure 7.5.Obviously triangulations under YCbCr and L*u*v colours models are more suitable for the used encoding technique (based on Huffman encoding).

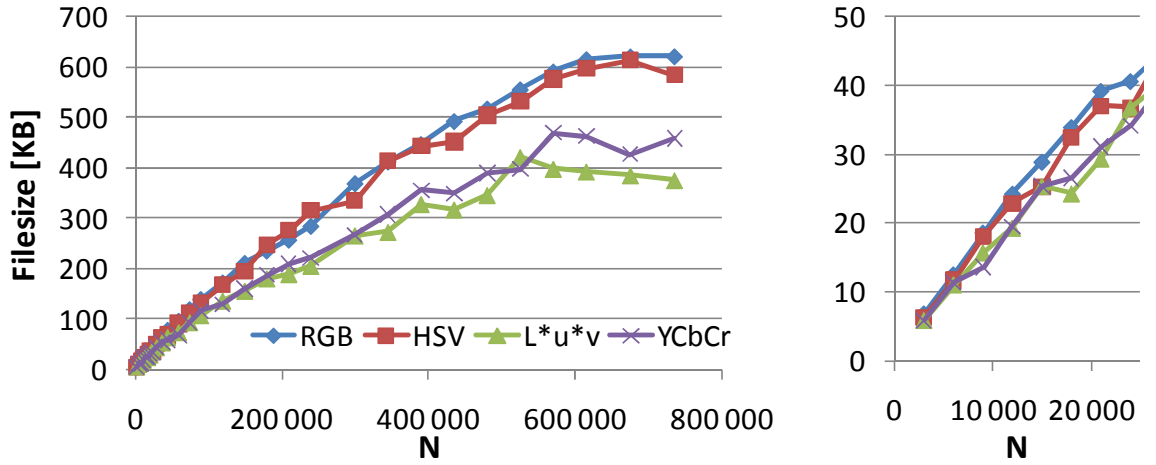


Figure 7.5: *The dependency of sizes of output files encoded by FVXPATH + bzip2 technique on the overall number of vertices for colour 512x512 fruits image.*

However, it is important to point out that the quality of the geometric representation using these colour space systems degrades faster and, therefore, we must take into account also this information – see Figure 7.6. Now, there is no doubt that the most suitable colour model is the YCbCr model followed by the RGB model unless we work with large triangulations for whose the RGB model is the best one.

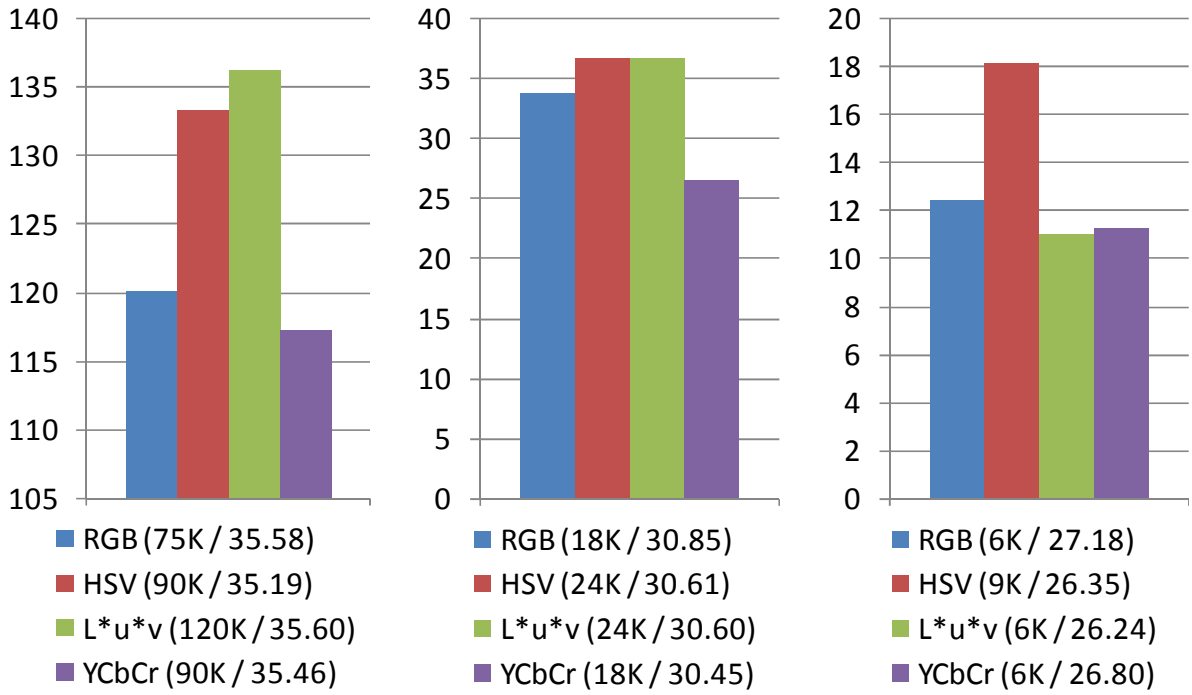


Figure 7.6: *The sizes of output files (encoded by FVXPATH + bzip2 technique) for three test cases of colour 512x512 fruits image. Information in brackets denotes the overall number of vertices in the triangulation (in thousands) / the PSNR of the representation.*

7.3 Cotriangulation

A drawback of the straightforward approach described in the previous section is that it introduces different approximation error for different colour components. It might results in colour bleeding. For an example, let us suppose to have a pixel of pastel pink colour with RGB values 235, 205 and 220 in the original colour image. Let us further suppose that whilst the red and green components are reconstructed without an error, the reconstructed value of the blue component is 180. While in a grey-scale image, an approximation error of this scale would be hardly spotted, in the considered colour image, it is well visible as the reconstructed pixel has a beige colour instead of ping one – see Figure 7.7.

Figure 7.7: *Illustration of colour bleeding – top: the original colour, its RGB components and the corresponding grey-scale value; bottom: the colour reconstructed with an error in the blue component.*

Colour bleeding can be diminished, if we use one cotriangulation [Wei98], instead of three separated triangulations. In general, the cotriangulation is constructed from *n* *d*-dimensional triangulations by their transformation into one *d*-dimensional triangulation in *(d+n)*-dimensional space. For the purpose of the triangular representation of colour image, this means that the inputs are three component triangulations and the output is one triangulation that includes all the colour channels. Hence, the resulting structure is the Delaunay triangulation with five parameters x, y, R, G, B assigned to each vertex (instead of three parameters x, y, grey value as in the case of grey-scale images).

As the spatial distribution of vertices in input triangulations usually does not match, i.e., for some vertex of one triangulation, there is no vertex at the same position in the other two triangulations, the number of vertices in the cotriangulation ranges from the vertex count in the smallest component triangulations to the sum of vertex counts of all input triangulations. An important property of the cotriangulation is that it might represent some vertices from the input triangulations with additional approximation error, which also reduces the storage costs. Detailed description of cotriangulations and their use for our purpose is given in the bachelor thesis by T. Janák [Jan09] and in the bachelor thesis R. Sýkora [Syk08].

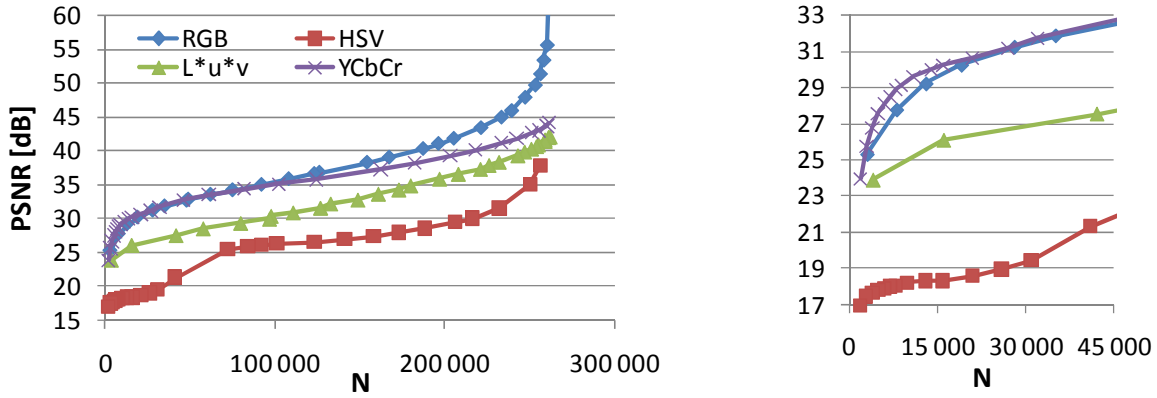


Figure 7.8: *The dependency of quality (measured in PSNR) of images reconstructed from the Delaunay cotriangulation on the overall number of vertices (in all component triangulations) for colour 512x512 Lena image.*

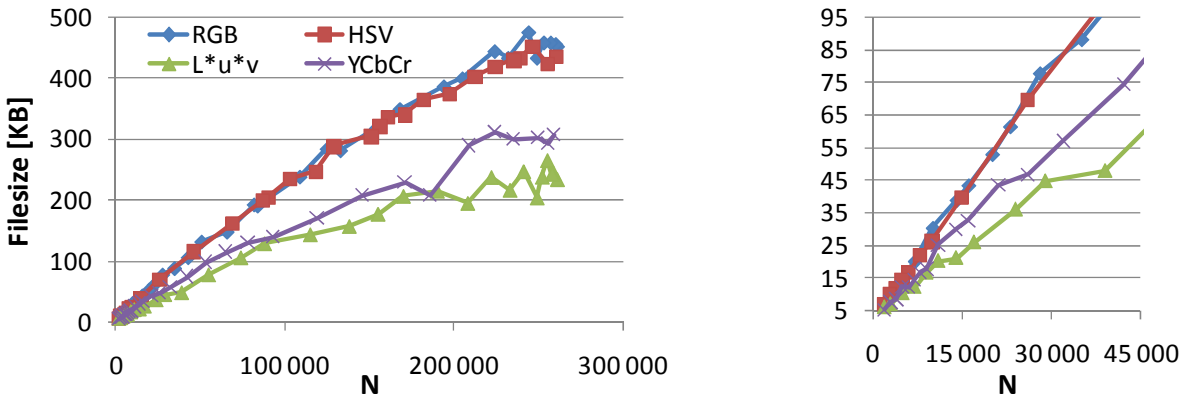


Figure 7.9: *The dependency of sizes of output files encoded by FVXPATH + bzip2 technique on the overall number of vertices in the Delaunay cotriangulation for colour 512x512 fruits image.*

Figure 7.8 and Figure 7.9 show the dependence of the quality (measured in PSNR) of the representation by the cotriangulation on the number of vertices in this cotriangulation and the relationship between the number of vertices in cotriangulations and the sizes of files produced by FVXPATH + bzip2 method extended for cotriangulations (instead of one array of grey values, three arrays are stored – one for each component) and. As it can be seen the curves have similar trends as curves for separate triangulations (see Figure 7.4 and Figure 7.5) with a small exception of L*u*v colour model whose quality degrades in cotriangulations faster.

The dependency of quality on output file sizes is given in Figure 7.10. It is obvious that YCbCr colour model outperforms the other colour models, although, save for the HSV model, differences are not big. Figure 7.11brings another comparison. Supposedly, L*u*v model provides us with such good results that it could be considered as the second best model. However, this model was found unstable; it may perform poorly (see Figure 7.12) for some images. Hence, the most suitable model is the YCbCr model followed by the RGB model.

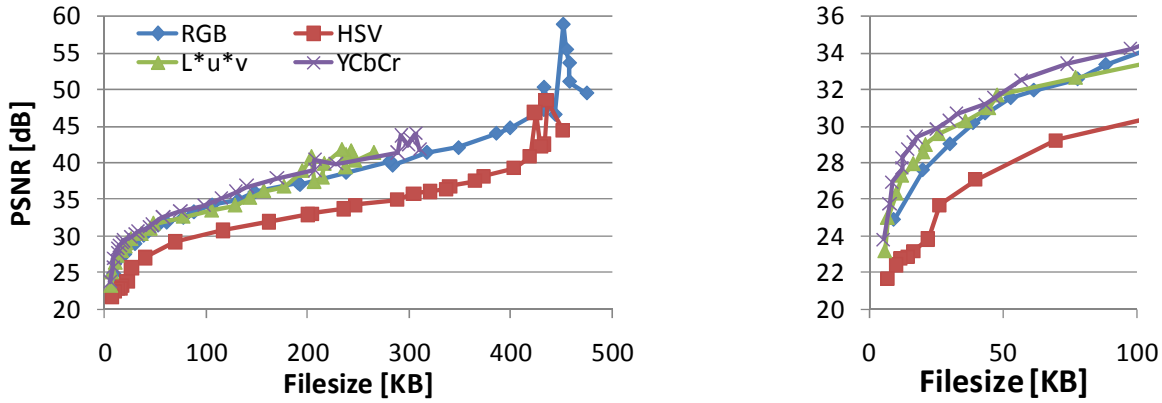


Figure 7.10: *The dependency of quality (measured in PSNR) on sizes of output files with cotriangulations that were encoded by FVXPATH + bzip2 technique for colour 512x512 fruits image.*

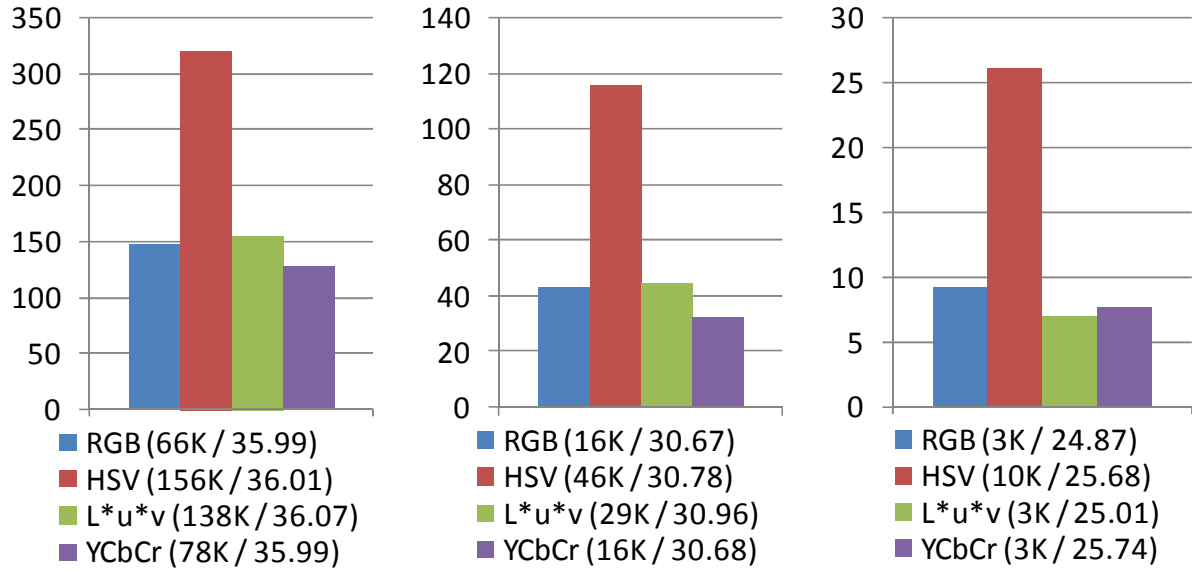


Figure 7.11: *The sizes of output files (encoded by FVXPATH + bzip2 technique) for three test cases of colour 512x512 fruits image. Information in brackets denotes the overall number of vertices in the cotriangulation (in thousands) / the PSNR of the representation.*

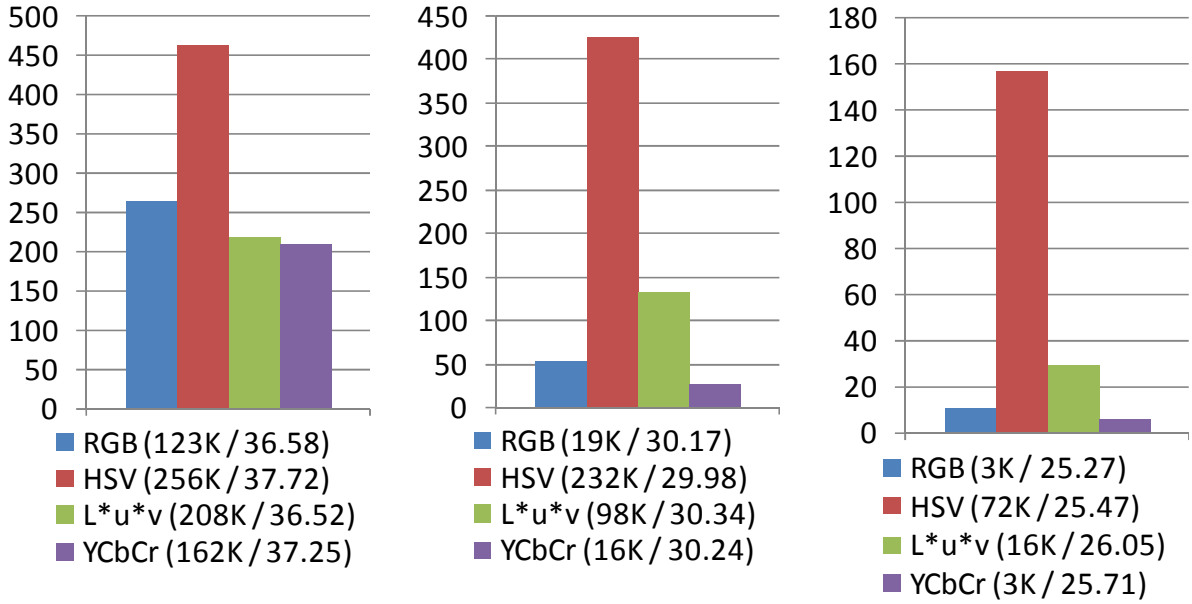


Figure 7.12: *The sizes of output files (encoded by FVXPATH + bzip2 technique) for three test cases of colour 512x512 Lena image. Information in brackets denotes the overall number of vertices in the cotriangulation (in thousands) / the PSNR of the representation.*

7.4 Comparison & Summarization

When we compare separated triangulations with cotriangulations (see Figure 7.13), we can see that separated triangulations outperforms cotriangulations in case of RGB model, whilst in case of YCbCr model the situation is vice versa. The improvement of cotriangulations is, however, negligible considering the increased complexity of the algorithm. Hence, we recommend the use of separated triangulations (three) for the representation of colour images and these triangulation should exploit YCbCr colour model, when low bit rates are demanded, otherwise RGB model is more suitable.

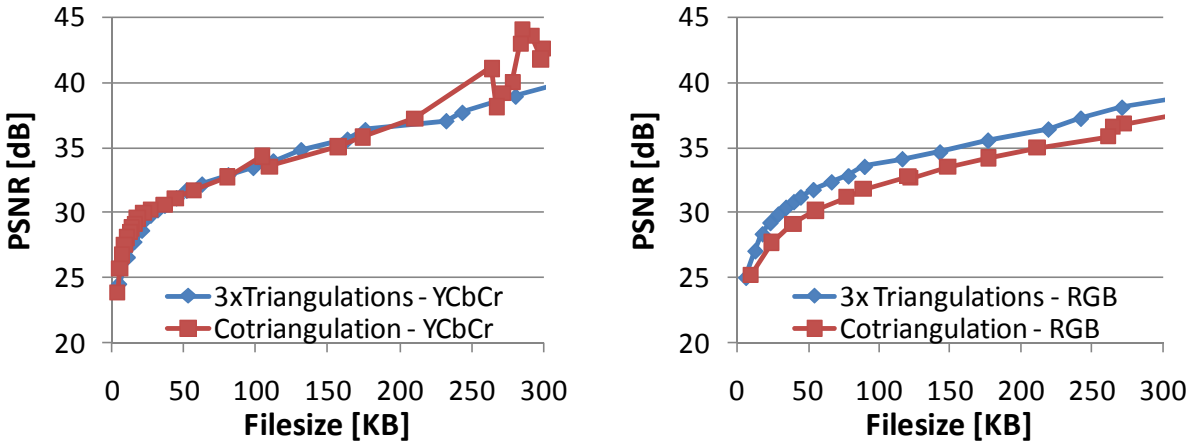


Figure 7.13: *Quality comparison of separated triangulations with cotriangulations for colour 512x512 Lena image.*

Detailed description of performed experiments can be found in thesis by R. Sýkora [Syk08].

8 Extension for Video

A digital video is a sequence of usually similar images called frames. The most widely used representation of video is based on the partition of these frames into groups containing intra-coded *I*-frames, inter-coded *P*-frames and optional bidirectionally inter-coded *B*-frames – see Figure 8.1. Each *I*-frame is considered to be a standalone image and is processed independently, i.e., it is encoded without using any information from the previous (or the following) frames in the video sequence. For the encoding, the majority of algorithms for lossy video compression used nowadays exploits either the discrete cosine transform (MPEG1-2, DV, MJPEG, H261-4), the discrete wavelet transform (MJPEG 2000, Intel Indeo 5) or the vector quantization (Cinepak, Sorenson Video).

The inter-coded and bidirectionaly inter-coded P-frames and B-frames exploit their similarity to the surrounding frames (in one or both directions) for the encoding. In most cases, some block matching algorithm is used. For each macroblock in the frame (typically, 16×16 pixels group of four 8×8 pixel blocks – a basic element for JPEG based compression), it searches for such movement vectors that, if applied to the macroblock, describe the following frame with the minimal error. The process is illustrated in Figure 8.1 that shows a block *B* that has been found to be similar to the block *B'* in the same window in the next frame.

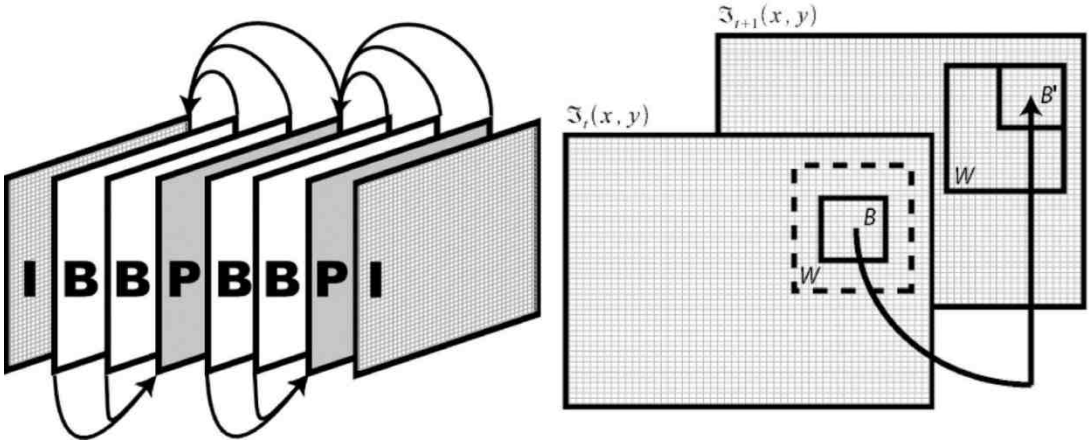


Figure 8.1: *Typical compression scheme (left) and block matching principle (right).*

The most common problems connected with traditional video compression schemes (described above) are the appearance of new block elements between two frames and a serious loss of detail at low bitrates. Moreover, when there is a need to transform or interpolate a videosequence, handling these video representations is a bit impractical as the corresponding frames have to be decompressed and all their pixels processed. Both drawbacks could be solved (or at least diminished) by using an alternative triangulation-based representation.

A straightforward approach would be to represent intra-coded frames by the Delaunay triangulation (see Section 3, 4) of subset of important pixels, encoded into a compact form by some of our encoding methods (see Section 5, 6), and process inter-coded frames in a traditional way (like in MPEG). In our research, we investigated two other options of exploitation of triangulations in a digital video. First one uses the kinetic Delaunay triangulation where vertices move in time (in directions computed by a block matching algorithm). The other one is based on the idea that video could be represented by 3D triangulation, a tetrahedrization, since it can be considered as a 3D matrix of pixel data taking the time axis as the third geometrical coordinate.

8.1 Kinetic Delaunay Triangulation (KDT)

The idea of video representation by the kinetic Delaunay triangulation is based on a creation and successive movement of the Delaunay triangulation.

8.1.1 Encoding

In the first step, the method computes the Delaunay triangulation of the most significant points obtained from a frame that is considered to be intra-coded. These points can be identified either by one of mesh based heuristics described in Section 3 (in our experiments we use the BRUTE heuristics) or by a meshless heuristics specially developed for the purpose of video encoding that works as follows.

First of all, points lying on image edges are detected using some of existing edge detectors followed by a threshold operator, which filters out less important points. We experimented with the most commonly used detectors: Marr-Hildreth, Sobel, Robinson, Roberts and Prewitt. Apparently, the Marr-Hildreth operator, which is also known as Laplace operator, is the best choice (in the meaning of the achieved quality) – see Figure 8.2. Next, some random points are chosen and these points together with points obtained in the first step are combined together to get the resulting set of significant points. These points then define the KDT until a new picture group is formed, starting with the next intra-coded frame.

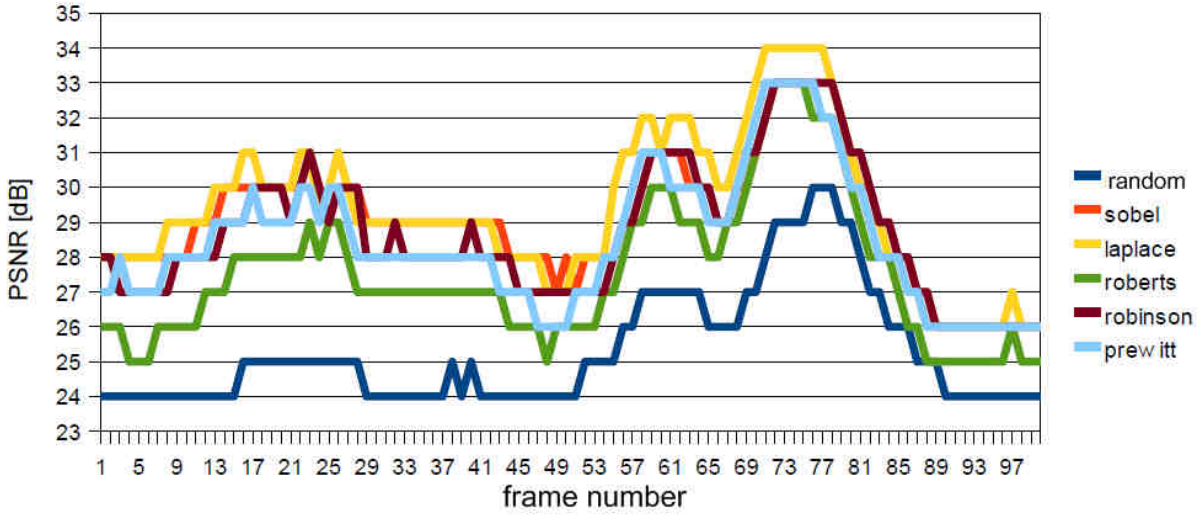


Figure 8.2: *The quality of the representation by the Delaunay triangulation of 20% points in the dependence on the used edge detector for the Foreman video. No inter-coding is used.*

The optimal ratio between edge and random points was investigated and the results are summarized in Figure 8.3. It can be seen that in the case of live video with dynamic camera, where both background and foreground changes (Foreman), the optimal ratio is about 60:40, while in the case of live video with static camera, where background is simple and do not change (Miss America), the ratio is much higher, it is about 80:20. Rendered video without noise is best represented by the ratio 50:50. Hence, a good compromise seems to be the ratio 50:50, i.e., half of significant points are chosen on image edges and half are chosen randomly.

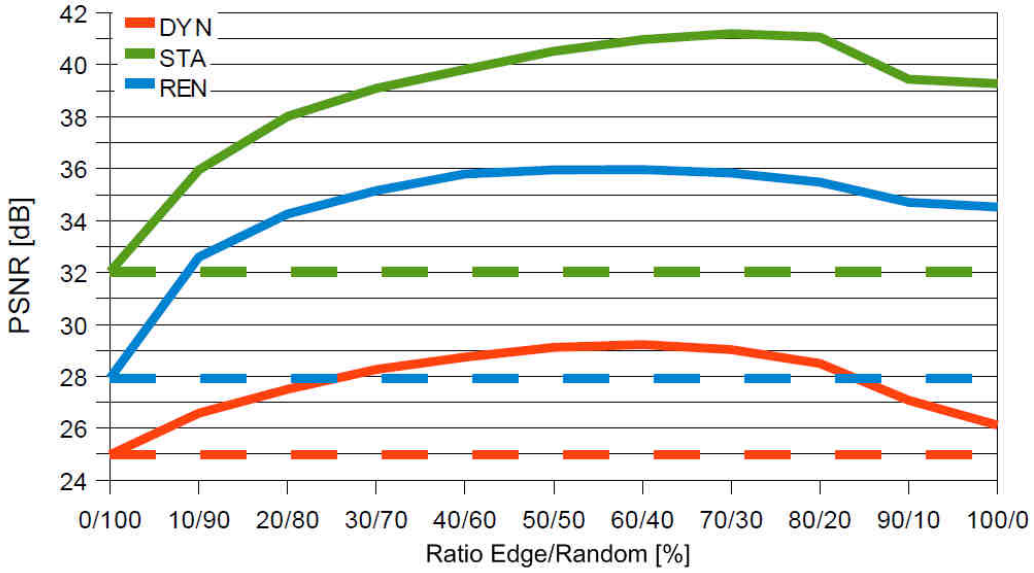


Figure 8.3: *The quality of the Delaunay representation in the dependence on the ratio of edge and random points to be triangulated for three videosequences – live video with DYNamic camera, live video with STAtic camera and RENdered animation. The overall amount of selected points is 20%.*

For inter-coding (i.e., *P* and *B* frames), methods for optical flow prediction are used [Hor81]. We experimented with both the block matching algorithm (BMA) [Fur97] using three different metrics: Mean Square Difference (MSD), Mean Absolute Difference (MAD) and Pel Difference Classification (PDC) and the differential method Kanade-Lucas-Thomasi (KLT) [Luc81]. According to the performed experiments [Pun08], the block matching with MAD metrics outperforms the KLT method since the latter method detects movement in the triangulated image even in static places – see Figure 8.4. In the further text, the BMA with MAD will be assumed, if not explicitly expressed otherwise. Additional investigation showed that optimal macroblock size is 16×16 pixels and, therefore, the size of searching window (see *W* in Figure 8.1) is 31×31 pixels (it corresponds to the largest possible movement). Let us note that we used five inter-coded frames per one intra-coded frame.

Figure 8.4: *Motion vectors obtained from BMA (left) and KLT (right) methods.*

Once all the motion vectors in a frame are known, velocities of their points in the KDT are set, which gives us a new KDT – see Figure 8.5. An advantage of this approach is that, unlike the traditional pixel based methods, the movement of a block (triangular) of image is continuous, i.e., a better quality can be expected when the frame rate of the considered video changes.

On the other hand, time consumption is significantly larger and the straightforward implementations of KDT are known to suffer from the numerical instability. A robust Kinetic Delaunay Triangulation is described in [Vom08, Vom09].

Figure 8.5: *The original KDT, detected motion vectors and the movement compensation in the KDT.*

As the extension for video was developed in parallel to the development of encoding techniques that were presented in Section 5, the KDT is encoded using other, not so powerful, methods. The employment of the FVXPATH or other, more suitable, method belongs to the scope of our future research. At present, intra-coded frames are encoded by a modified the RAW method that sorts all points according to the grey values associated with them and then, starting from the second point, it replaces the grey value of each point by the difference in grey values between this point and the previous one. Five bytes are required to represent one point (coordinates require two bytes). The final stream is stored into the output file and further compressed by the deflate compression technique (see Section 5). In the following text, this encoding method will be denoted as DELTA+GZip. The block schema of the proposed technique for intra-coded frames is given in Figure 8.6.

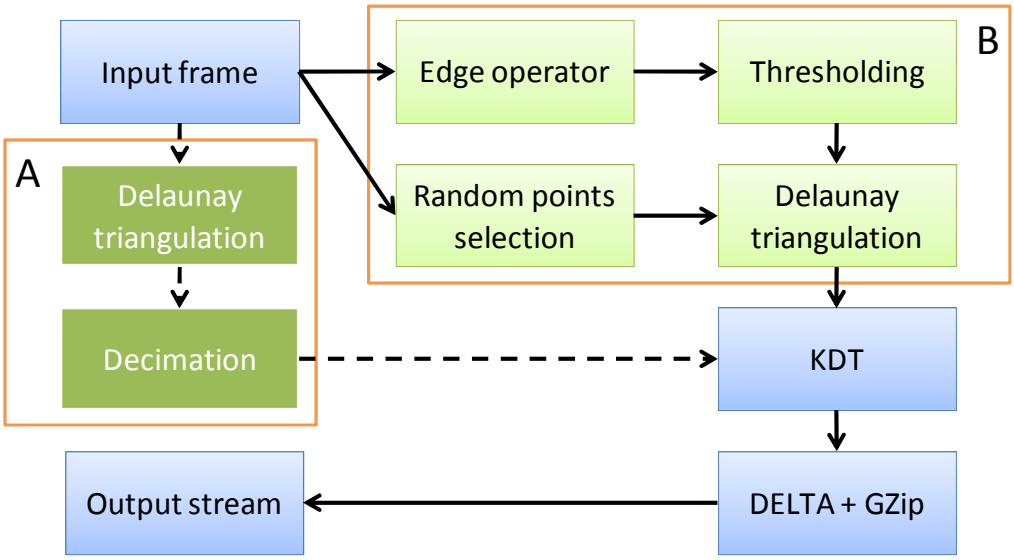


Figure 8.6: *Encoding of intra-coded (I) frames. Blocks A and B are mutually exclusive.*

For inter-coded *P*-frame, one motion vector is stored in a raw format for every point (vertex) in the previous intra-coded frame. As the movement is small, one byte per one vector component is more than sufficient. The output file is further compressed by the deflate compression technique. In the following text, this method will be denoted as RAWM+GZip. The block schema of the proposed video encoding is given in Figure 8.7.

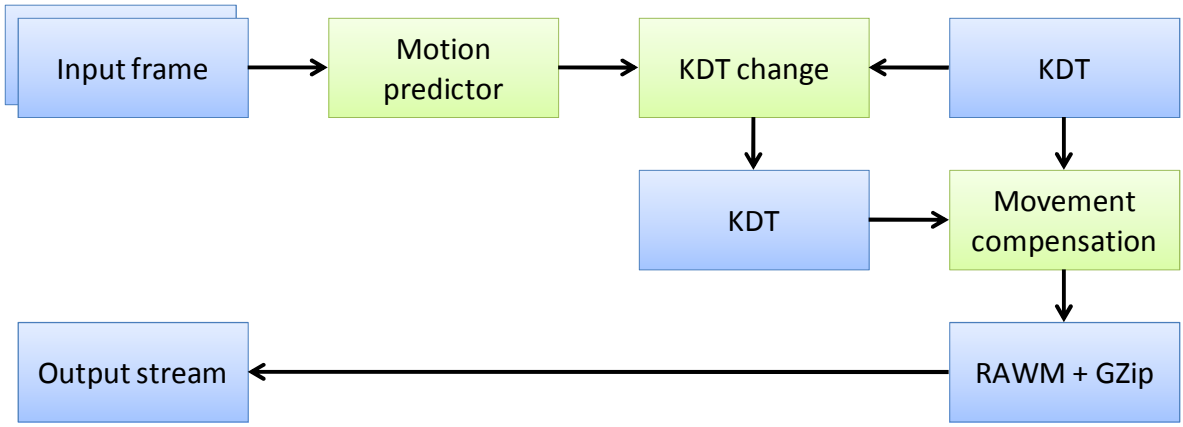


Figure 8.7: *Encoding of inter-coded (P) frames.*

8.1.2 Decoding

As the topology is not retained, to decode an intra-coded frame, points have to be retrieved from the input stream and the Delaunay triangulation of these points must be computed first. Triangles are then interpolated by the bilinear interpolation. In the case of inter-coded frames, we dynamically move vertices of the Delaunay triangulation according to motion vectors retrieved from the input stream. After that, either the triangulation can be interpolated like in the case of intra-coded frames or feature based warping may be applied. The warping process takes the edges of a triangle in the current frame and corresponding transformed edges in the previous frame. The task is to compute grey values of the pixels within the transformed triangle – see Figure 8.8. We adopted the warping for more line pairs as described in [Žar98]. Let us note that as we have a relatively accurate approximation of the grey values of all the pixels in the last intra coded frame, we perform the warping process for all the consecutive inter-coded frames after that frame. An expected advantage of the warping process over the interpolation is that if a triangle changes significantly, details are better preserved. The block schema of the overall decoding process is depicted in Figure 8.9.

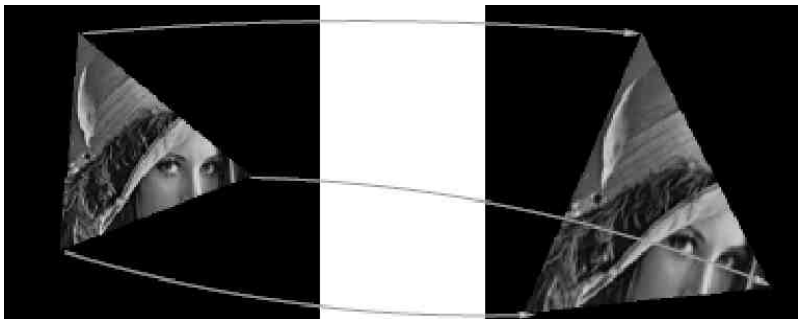


Figure 8.8: *Triangle warping [Mar00].*

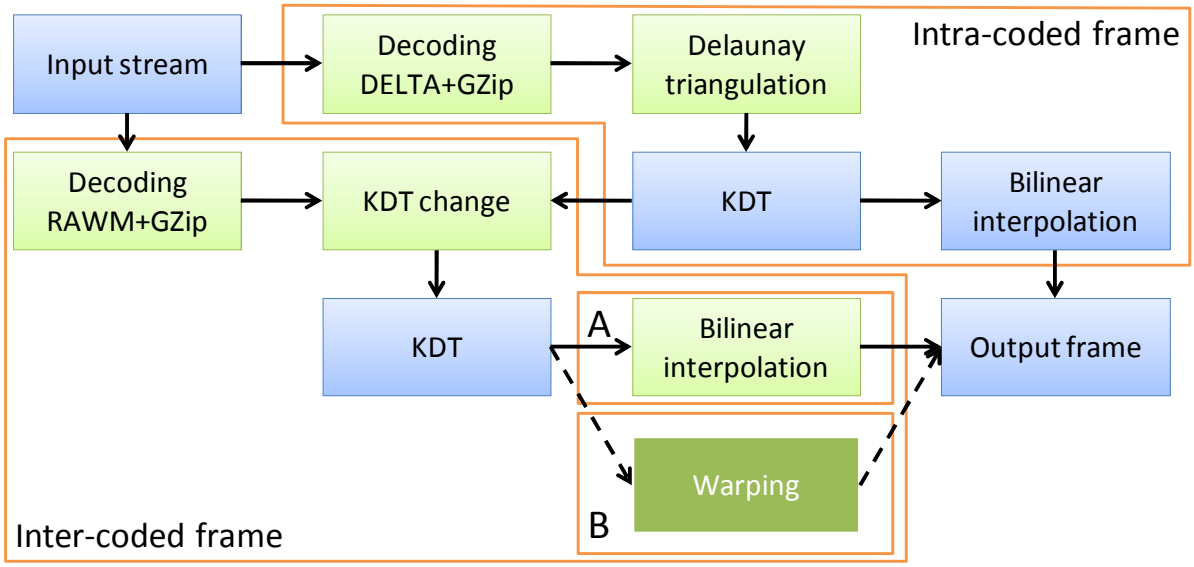


Figure 8.9: *Decoding of intra-coded (I) frames and inter-coded (P) frames. Blocks A and B are mutually exclusive.*

8.1.3 Experiments and Results

The proposed encoding achieves the compression ratio 20:1 – 4.5:1 [Pun08] for the tested videos (both real live videos and rendered animations were subject to experiments). Figure 8.10 brings a comparison of the quality of the proposed method with XviD for the same coded output size. The initial amount of inserted points was 5% and the length of the pictures group was set to 6. While the intra-coded frames provided us with the reasonable quality (although with not as good as XviD did), the inter-frames quality dropped rapidly. Both interpolation and warping techniques were measured to be nearly equivalent. A subjective comparison, however, often prefers the warping prior to the interpolation since it reduces the number of triangular artefacts which may appear as a result of the reconstruction.

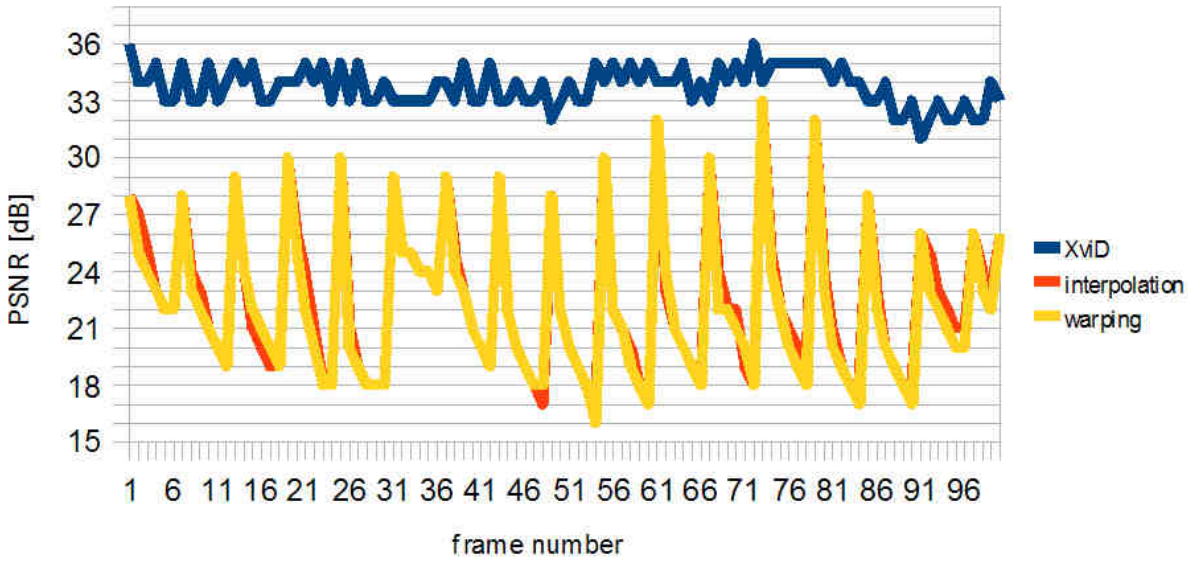


Figure 8.10: *Quality comparison of the proposed method with XviD.*

When the BRUTE method was used for the selection of important points, the quality improves significantly – see Figure 8.11. The improvement measured in PSNR is about 6 dB, which, especially, if a more powerful encoding method were used (see Section 5), could be enough to make the proposed intra-coding an alternative to XviD encoding. Let us note that the BRUTE method, however, requires double time compared to the default meshless heuristics described in this section [Pun08].

An introduction of some kind of nonlinear movement (for instance the movement along elliptic trajectories) may bring improvement for inter-coding, which is thoroughly needed to render triangular representations of video useful. The research is, however, still in progress.

Figure 8.11: *Comparison of the image reconstructed from the Delaunay triangulation of 20% points selected by proprietary refinement technique (left), the BRUTE decimation technique (middle) with the image that was subject to M-JPEG compression.*

Detailed description of the method and presentation and analysis of other results of performed experiments can be found in thesis by P. Puncman [Pun08].

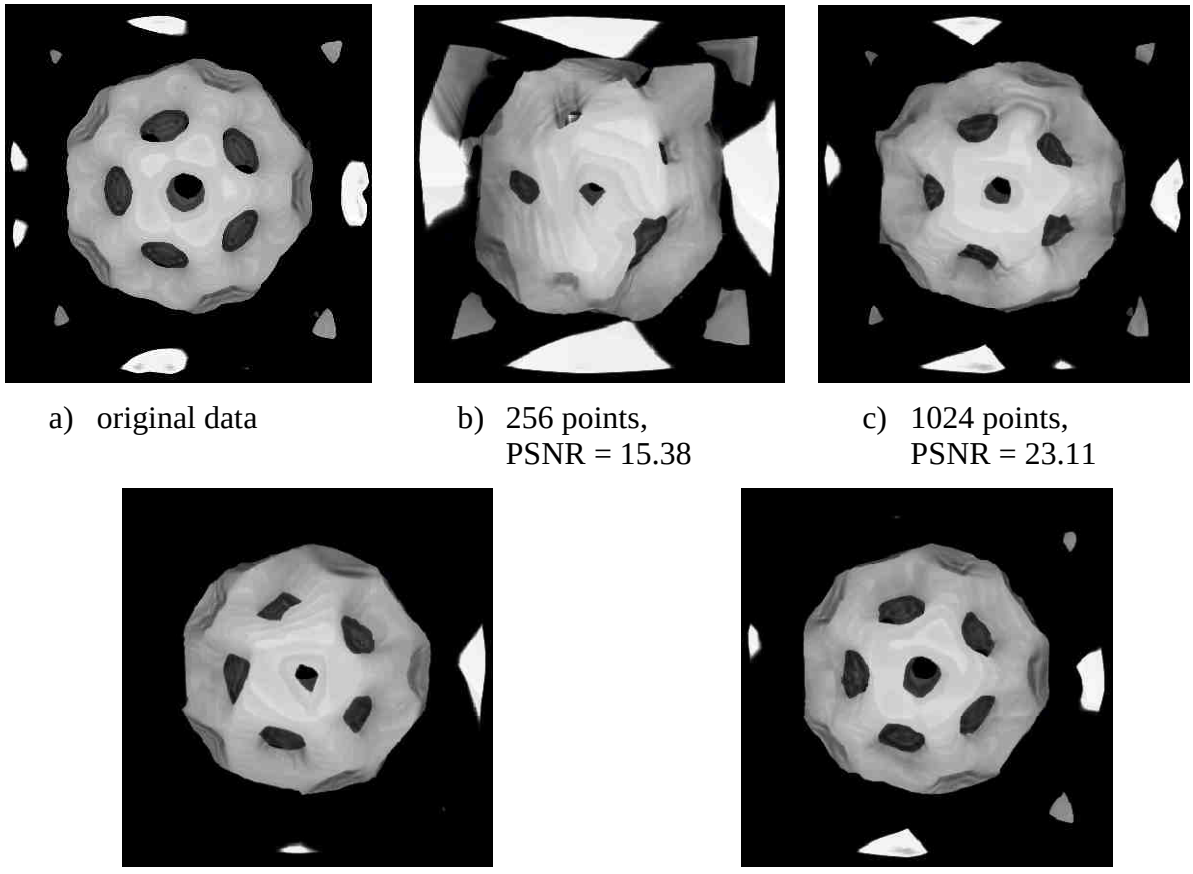
8.2 3D Delaunay Triangulation

A video sequence represented by a set of frames of $N \times M$ pixels can be transformed (without any loss of information) into a point set S in E^3 such that for each pixel there is one point with z-coordinate defined as a function of the frame number, x and y-coordinates defined by the position of this pixel in the framer and with the associated data corresponding to the grey-scale value (or colour components values) of this pixel. The key issue is to determine the relationship between units in x-axis (or y-axis) units in z-axis, i.e., if the distance between two pixels adjacent in one frame is one, what is the distance between a pixel in one frame and its corresponding pixel in the adjacent frame? For the sake of simplicity, we decided to assume that this distance is also one, i.e., z-coordinate of the point corresponds to the frame number. In our future research, we would like to perform experiments to find the correct relationship.

Starting with the initial 3D Delaunay triangulation formed by an artificial tetrahedron large enough to contain all input points, points are successively added into this triangulation in the order such that the next point to be inserted is the one that participates most at the total error of the approximation (again bilinear interpolation is used for the reconstruction). The insertion is repeated until either the maximum allowed error is achieved or the maximum allowed number of points is inserted. In order to speed up the process, several techniques were adopted, e.g., insertion of several points simultaneously – see [Var07].

Let us note that the produced Delaunay triangulation can be improved (in the meaning of the quality of the representation) by a simulated annealing. As it can be seen in Figure 8.12, this is especially true for small triangulations. A significant drawback of the simulated annealing is its enormous time consumption. While the triangulation of 256 points took 36 seconds, its improving by the simulated annealing needed more than one hour. As a video sequence is

typically represented by larger triangulations, for whose the improvement brought by the simulated annealing is not substantial, we opted out using this strategy for video encoding.



a) original data

b) 256 points, PSNR = 15.38

c) 1024 points, PSNR = 23.11

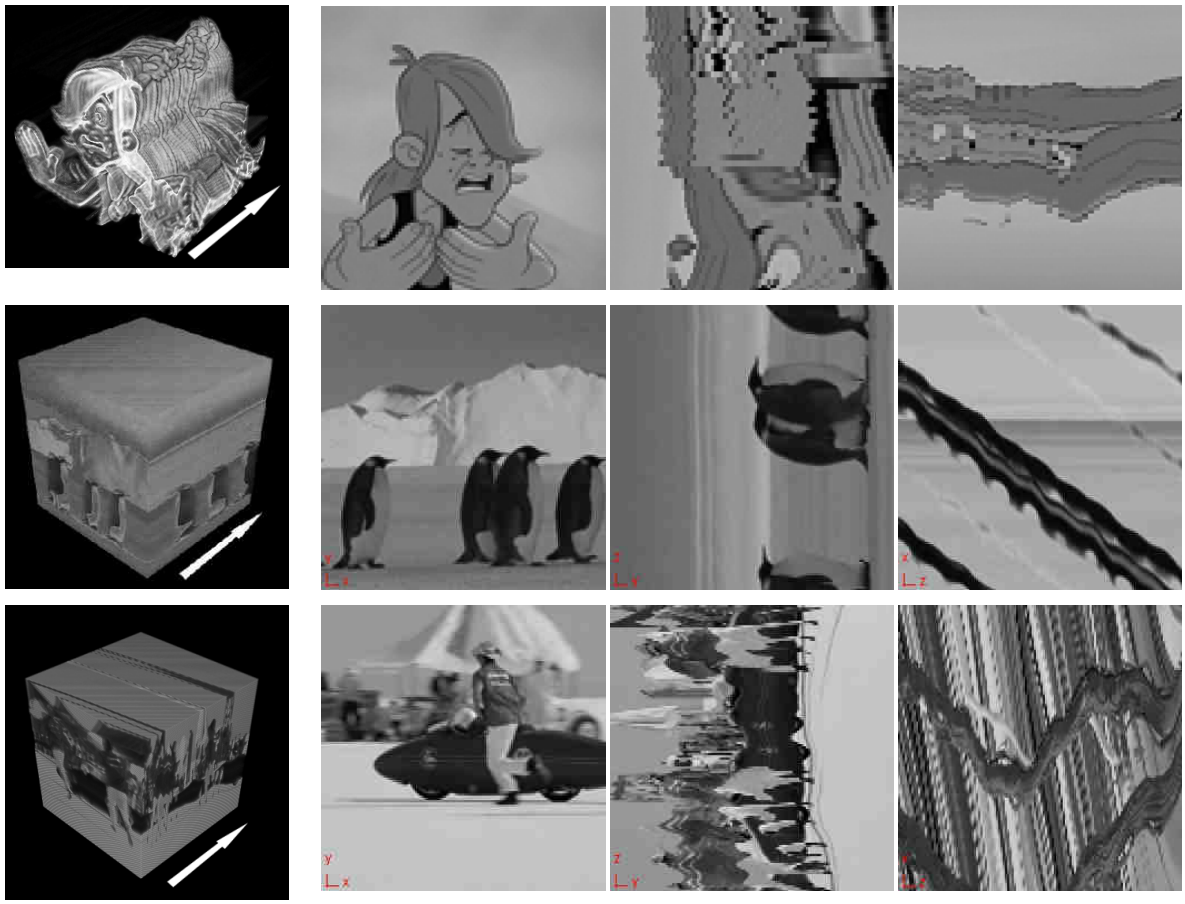
d) 256 points, simulated annealing, PSNR = 22.79

e) 1024 points, simulated annealing, PSNR = 26.25

Figure 8.12: *Quality comparison of volumetric data reconstructed from the 3D Delaunay triangulation of 256 and 1024 points constructed without and with simulated annealing. Buckyball 32x32x32.*

The extension for video was developed in parallel to the development of encoding techniques that were presented in Section 5 and, therefore, the 3D Delaunay triangulation is encoded using another, not so powerful, method, which was found to be the best one from several tested methods [Var07]. This method, denoted as VarC+GZip, sorts all vertices of the triangulation according to x, y and z-coordinates (the order is not important), computes differences in coordinates between adjacent points and stores differences in x-coordinates followed by differences in y-coordinates and by differences in z-coordinates and followed by grey values into the output file. Due to the sorting, differences should be small and, therefore, more suitable for the deflate compression algorithm that is used on the output file to reduce its size.

The approach described above was tested on three video sequences: a cartoon and videos with the static and the dynamic camera. Figure 8.13 shows these videos as 3D volume data rendered by the ray-tracing technique and sliced in various axes. It can be seen that a typical cartoon scene is very coherent; a character moves on some usually not too much variable background. On the other hand, most of edges are very distinct, sometimes even with a black contour. While the inner structure of the video produced by a static camera is also quite simple, the structure of dynamic camera video is complex and unlikely to be well compressible.

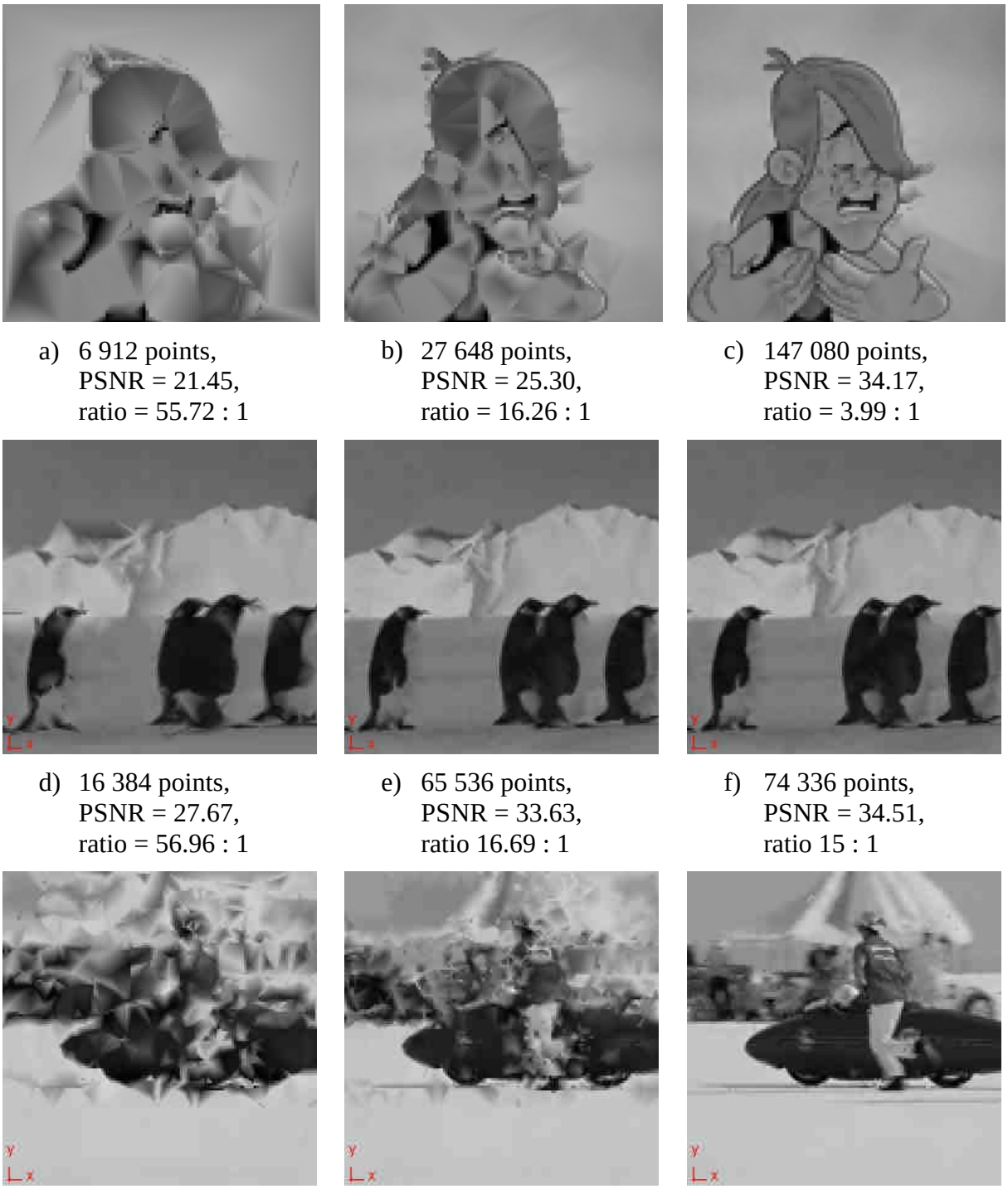


a) ray-tracing b) slices of the video data in z-axis, x-axis and y-axis

Figure 8.13: *Cartoon, static and dynamic camera video sequences viewed as volume data.*

Figure 8.14 brings a comparison of compression ratio achieved for the tested videos. From the tested types of data, the video produced by a static camera is apparently the most proper for our proposed method, while the video with dynamic camera is improper. Surprisingly, the cartoon type of video was also not a winning story. All in all, results show that the proposed method does not fit the inner structure of the video well since the three axes are not of the same character and, thus, they should be handled differently. Moreover, the interpolation on long tetrahedra (slivers), which are present in the final 3D Delaunay triangulation in not insignificant count, blurs edges, which has a negative influence on the quality of the represented image. Some improvement could be achieved by an incorporation of faces and edges as constraints into the triangulation but this would bring unlikely a substantial improvement. Hence, we can summarize our research in this field as follows. Although the alternative representation of video by 3D Delaunay triangulation is possible and theoretically interesting, it is useless from the practice point of view as it cannot compete with the traditional approaches.

Details can be found in the master thesis by M. Varga [Var07].



a) 6 912 points, PSNR = 21.45, ratio = 55.72 : 1

b) 27 648 points, PSNR = 25.30, ratio = 16.26 : 1

c) 147 080 points, PSNR = 34.17, ratio = 3.99 : 1

d) 16 384 points, PSNR = 27.67, ratio = 56.96 : 1

e) 65 536 points, PSNR = 33.63, ratio 16.69 : 1

f) 74 336 points, PSNR = 34.51, ratio 15 : 1

g) 16 384 points, PSNR = 18.21, ratio 56.91 : 1

h) 65 536 points, PSNR = 22.53, ratio 16.66 : 1

i) 512 011 points, PSNR = 34.36, ratio 3 : 1

Figure 8.14: *Compression ratio achieved for the tested video sequences.*

9 Triangles Interpolation

The quality of the image reconstructed from the geometric representation, more specifically from a triangulation, is undoubtedly influenced by the way how the triangulation is interpolated to get the missing pixels. Although Dyn et al. showed in [Dyn90] that a piecewise bilinear interpolation on a Data-Dependent Triangulation (DDT) can lead to plausible results, this simple approach obviously does not generate satisfactory results in a general case, e.g. for an arbitrary triangulation. Main problems are that large, almost monotonous, areas are not smooth enough whilst image edges are not sharp enough. In this section, we investigate various interpolation techniques for triangular meshes.

Without any doubt the simplest and also the fastest interpolation method is the constant interpolation that assigns the average of grey-scale values held by vertices of a triangle to every pixel of this triangle. A comparison of this interpolation with commonly used bilinear interpolation, which was also considered to be the base of our previous experiments, is given in Figure 9.1. It can be seen that although the bilinear interpolation does not produce as visible triangular artefacts as the constant interpolation, it is still not perfect. Edges of interpolated triangles are clearly identifiable in particular places of the reconstructed image – see Figure 9.1d. These flaws are apparently caused by an ignorance of intensity behaviour in areas surrounding the interpolated triangle.

a) triangulation b) constant interpolation c) bilinear interpolation

d) bilinear interpolation – detail of the reconstructed image 1024×1024

Figure 9.1: *The reconstruction of Lena image from the Delaunay triangulation with 4 000 using constant and bilinear interpolations on triangles.*

Options which would enable us to incorporate those areas into calculations of the resulting intensity can be differentiated into two main groups. First are methods, which still interpolate the triangles individually, but use vectors respective to the continuous intensity surface varying across the whole image to correlate the output. The other approach is to interpolate on more complex surface structures formed by individual triangles of the triangulation.

Zienkiewicz interpolation is an alternative for the bilinear interpolation. It uses gradient vectors to describe the behaviour of the intensity in the area that surrounds the interpolated triangle. Gradient in a vertex can be estimated as an average of normalized surface normal vectors of each of the triangle adjacent to the vertex, weighted by their areas. Zienkiewics interpolation is a bit slower than the bilinear interpolation, however, for common applications, this slow down can be neglected.

The Bezier and Coons patch methods interpolate small overlapping surfaces formed by a couple of adjacent triangles. Their advantage is that they successfully remove traces of the triangulation, making the image smoother than in case of bilinear or Zienkiewicz interpolation. However, this also introduces an unwanted blending of colour edges in the image, which might lead produce artefacts – see Figure 9.2.Therefore, these methods are not suitable as universal interpolation methods. Nevertheless, this approach could be exploited more and used for partial interpolation of smooth areas or for interpolation of specially prepared triangulations that would ensure that the artefacts would not appear. Another drawback of these methods is an extremely large computational time, which renders these methods useless.

Figure 9.2: *Part of the Fruits image, interpolated by the Coons patch method from triangulations with 6000 vertices in red, 9000 vertices in green and 9000 vertices in blue component.*

The last tested approach exploits a dual configuration of the Delaunay triangulation, the Voronoi diagram. Whilst the piecewise linear interpolation over Voronoi cells proved to be a complete failure (see Figure 9.3a), the natural neighbour interpolation, which is based on measuring change of areas of Voronoi cells when the interpolated pixel is inserted into the diagram, brought some promising results –see Figure 9.3b. It requires a significantly higher time than the bilinear or Zienkiewicz interpolation but this is still within reasonable bounds. Although the quality of natural neighbour interpolation does not outperform the Zienkiewicz interpolation, it is close enough to take it into account in the further development.

a) piecewise linear interpolation over Voronoi cells

b) natural neighbour interpolation

Figure 9.3: *Lena image reconstructed from the Delaunay triangulation of 10 000 vertices.*

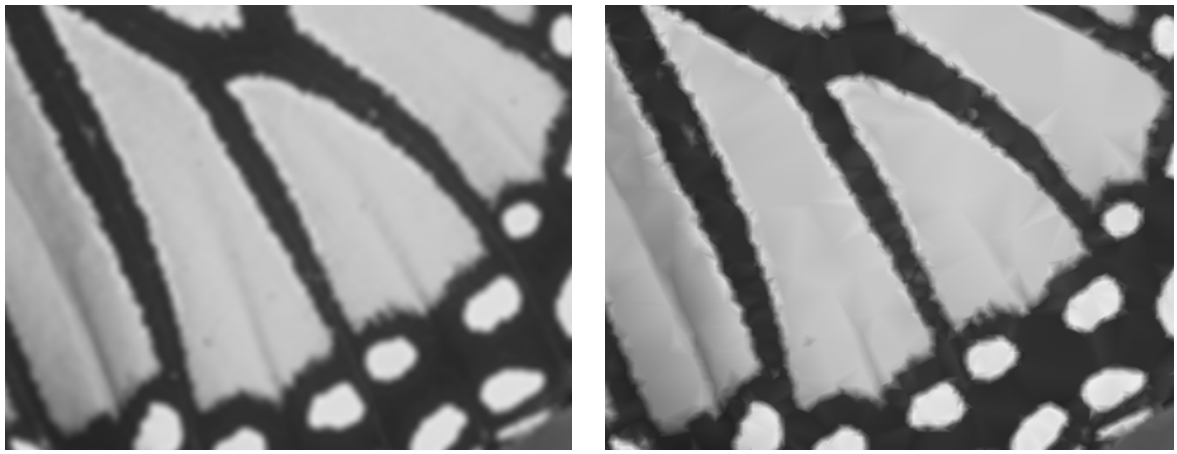
We experimented with both greyscale and coloured images. Coloured images may be represented by either separate triangulations for each colour component or the co-triangulation. Co-triangulations are more suitable when a more effective compression is required but their visual quality is usually worse. On the other hand, they avoid the problem of colour leaking, which is present in the case of separate triangulations (see Figure 9.2).

Detailed description of interpolation methods and performed experiments can be found in bachelor thesis by T. Janák [Jan09] and also in paper [Jan08].

10 Direct Manipulation with Triangulated Images

Real images captured by digital cameras are often unsuitable for many applications and some image enhancement techniques, such as resizing, change of contrast or brightness, gamma correction and smoothing of edges, must be applied. These techniques are well established for images represented by raster of pixels but, as far as we know, their extension for images represented by triangulations has not been discussed in literature. Indeed, it is always possible to perform the required enhancement operation in the reconstructed raster image and after that to compute a new triangular representation of the image but this straightforward approach brings two drawbacks. First of all, the transformation from raster form into geometric one takes some time and, therefore, it might be impractical to use this strategy always. More importantly, it could lead to severe degradation of quality since the considered transformation is lossy one, i.e., an information loss is present in every transformation.

Scaling of the image represented by the triangular mesh is the simplest operation. All that is needed is to apply the scaling to the vertices of triangles and then interpolate triangles. Even, if the Delaunay triangulation is used, instead of the Data Dependent Triangulation, which is more suitable for representation of images that should be scaled (see Section 2.4), the achieved visual quality can be better perceived than the visual quality obtained from the commonly used bicubic resampling of raster images. As it can be seen in Figure 10.1, the image reconstructed from the scaled Delaunay triangulation is sharp (in comparison with bicubic resampling) but on the other hand, some triangular artefacts are visible. Let us note that the visibility of these artefacts could be reduced in the post-processing by smoothing.



a) bicubic resampling

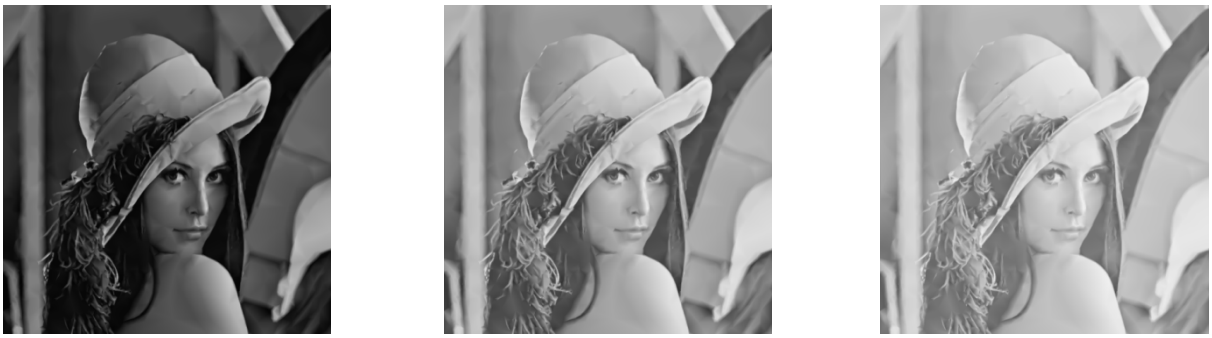
b) Delaunay triangulation + bilinear interpolation

Figure 10.1: *8 times scaled a part of monarch wing.*

Change of brightness in vertices of triangulation followed by the interpolation brings results comparable with traditional approach. The behaviour of the operation is demonstrated in Figure 10.2. An example of gamma correction is given in Figure 10.3. Again, in our opinion, the behaviour is reasonable.

a) original b) double brightness c) half brightness

Figure 10.2: *Direct brightness change for 512x512 Lena image represented by the Delaunay triangulation of 4 000 vertices.*



a) $\gamma = 0.5$ b) $\gamma = 1.3$ c) $\gamma = 2$

Figure 10.3: *Gamma correction for 512x512 Lena image represented by the Delaunay triangulation of 4 000 vertices.*

Smoothing is the most complex operation that we considered in our experiments. Although the transformation from the raster form itself smoothes the edges, additional smoothing may be required. The proposed method processes vertices of the triangulation one by one. It searches for vertices in the neighbourhood Ω of the tested vertex p and sums their weighted grey values (or colours) to get a new value for the vertex p. This can be written as formula:

---

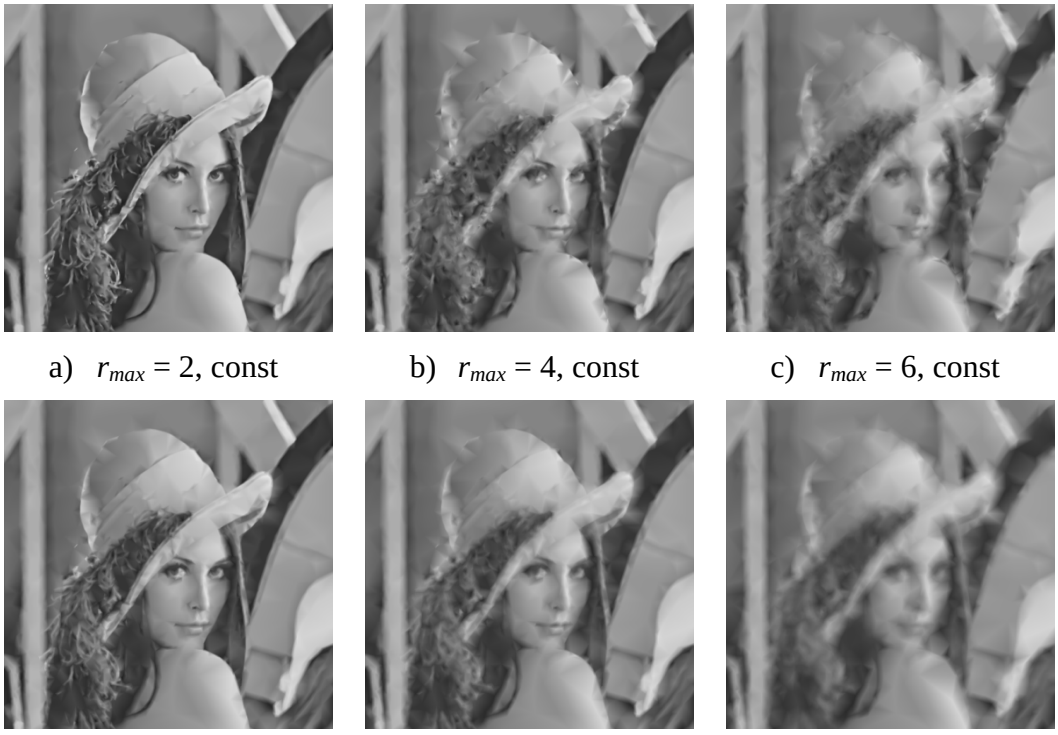
where denotes the grey values associated with the vertex . The weight can be either one, which means that all vertices have the same influence, or calculated as a function of the distance between the vertex p and the vertex q. We experimented with both Euclidian and topological distances. In the former case, the size of the neighbourhood Ω is defined by the circle of radius r_{max} given in pixel units. Naturally, a larger value of r_{max} leads into a smoother image. The following formula:

---

is then used to compute the weight. Note that weights decreases linearly with the Euclidian distance between vertices p and q.

If topological distances are considered, the size of the neighbourhood Ω is defined by the value d_{max} (again larger values mean smoother images). A vertex q lies in the neighbourhood of the vertex p only, if there is a graph path from the vertex q to the vertex p of at most d_{max} edges. The weight decreases linearly according to the number of edges d on the path; it is 1 for the most distant vertices, d_{max} for vertices connected to the vertex p.

Figure 10.4 displays results of the smoothing with Euclidian distances. When constant weights are used, the triangular artefacts occur in the image even for a very small neighbourhood size (compare the hat in Figure 10.4a and Figure 10.2a), whilst the image is still too sharp. Larger values bring a larger degree of smoothness but also more artefacts that makes the image noisy. Linear weights behave much better. The amount of artefacts is lower and the details are well smoothed as expected (see Figure 10.4e).



a) r_{max} = 2, const b) r_{max} = 4, const c) r_{max} = 6, const

d) r_{max} = 4, linear e) r_{max} = 6, linear f) r_{max} = 10, linear

Figure 10.4: *512x512 Lena image represented by the Delaunay triangulation of 4 000 vertices after smoothing by the method using Euclidian distances with constant and linear weights for various neighbourhood size r_{max}.*

Smoothing with topological distances is depicted in Figure 10.5. As it can be seen, the method does not introduce so many triangular artefacts as the method with Euclidian distances but on the other hand, the degree of smoothing is probably too big for typical applications. Even with the lowest possible neighbourhood size, the mouth is already too smooth no matter whether constant or linear weights are used. All in all, smoothing of an acceptable quality can be reached by repeating the smoothing by the method with Euclidian distances, linear weights and small neighbourhood size (e.g., 2 or 4).

a) $d_{max} = 1$, const | b) $d_{max} = 2$, const | c) $d_{max} = 1$, linear | *d)* $d_{max} = 2$, linear

Figure 10.5: *512x512 Lena image represented by the Delaunay triangulation of 4 000 vertices after smoothing by the method using topological distances with constant and linear weights for various neighbourhood size d_{max}.*

Apparently, it is possible to apply some image enhancement techniques (after some modifications, indeed) directly on the triangulated image with acceptable results. Especially, operations that deal with pixels independently (such as change of contrast or brightness, gamma correction or curves operations, change of colour hue or saturation, negation) can run without any unexpected problems. Further research in this area would be welcome but it is not planned as its benefit would be marginal.

11 Conclusion and Future Work

This report describes the most important results of the research project KJB10470701 (Alternative representation of image information by the use of triangulations) that was funded by GA AV of the Czech Republic in 2007 – 2009. It does not aim to be extensive and also it may not reflect last changes (especially, those in 2009).

In this report, we proposed and described various methods for the lossy transformation of raster of pixels into the Delaunay triangulation of the most significant points (corresponding to pixels) for both grey-scale and colour images. The majority of these methods start with a complete Delaunay triangulation of all points and successively remove points evaluated to be the least significant; a few of them start with an initial Delaunay triangulation and successively insert important points (see Section 3; 8.1.1). What makes them different is the heuristics used for the significance evaluation.

The best quality of representation is achieved by the BRUTE method (or one of its variants) that considers the point to be the least significant, if its removal from the triangulation would harm the quality least. If the image does not contain lot of edges, this method, when combined with powerful encoding techniques (see Section 5), outperforms JPEG in low bit rates and can more or less compete with JPEG2000,. On the other hand, the method is the slowest one; the transformation of 512x512 grey-scale image takes about one minute on a common hardware.

As for the colour images, given the required quality of representation, the best compression ratio is achieved, if the colour image is transformed into YCbCr or RGB colour model first and then each component processed separately as independent grey-scale image. YCbCr model is preferable for lower qualities, RGB for higher qualities. Cotriangulations proved to be not very useful for our purpose (see Section 7.3, 7.4).

Further experiments [Rul09] with the other triangulations (e.g., with weighted triangulations) proved that the differences in the quality achieved by the Delaunay triangulation and these triangulations are insignificant. As the Delaunay triangulation of a given set of points is unique (if some conditions are met) and, therefore, it can be encoded storing vertices (points) only and reconstruct from these vertices in the decoder, we consider the Delaunay triangulation (or CDT since the number of constraints is often limited) to be the favourite for the alternative representation of images.

Various methods for the encoding of the Delaunay triangulation were proposed (see Section 5, 8.1.1, 8.2) and tested. The FVXPATH+bzip2 method, which encodes the Minkowski differences in coordinates between two points adjacent in an array of points ordered in such a manner that the Minkowski differences are minimized, achieves the highest compression ratio. On the other hand, the method is quite time demanding. A good compromise between the compression ratio and the consumed time is brought by the BEHEC method, which performs a linearization of vertices using the Hilbert curve and then it stores the differences in their positions of this curve compressed by the Huffman encoding.

The raster image is reconstructed (in the decoder) from the Delaunay triangulation by an interpolation. From all interpolations we experimented with (see Section 9), the Zienkiewicz interpolation slightly outperforms the bilinear interpolation since it takes gradients in vertices into account. On the other hand, it needs slightly more time.

We also investigated the option of representing a digital video by the kinetic Delaunay triangulation (see Section 8.1) or by the 3D Delaunay triangulation (see Section 8.2). Although we do not consider the research in this area to be finished, we can conclude that while the representation by the 3D Delaunay triangulation sounds good from the theoretical point of view, it

is unlikely to be used in practice since the achieved compression ratio is far below compression ratios reached by traditional MPEG technique. The kinetic Delaunay triangulation behaves in a better way, however, due to its quality problems for inter-coding, it is also quite impractical. If inter-coding were improved, it could more or less compete with traditional techniques when low bit rates are required (e.g., in mobile phones broadcasting).

Direct application of common image enhancement techniques on the vertices of triangulation and its effect on the reconstructed raster image were discussed (Section 10). According to the experiments, operations that deal with pixels independently (such as change of contrast or brightness, gamma correction or curves operations, change of colour hue or saturation, negation) can run without any unexpected problems, other operations (such as smoothing) can be applied directly on the triangulated image with acceptable results.

Further research in this area should focus on a smart combination of triangulations and wavelet representations because, as it seems, triangulations itself are not sufficient enough to represent everything in the desired quality.

References

[Ada03] Adamian L, Jackups R, Binkowski AT, Liang J. Higher-order interhelical spatial interactions in membrane proteins. Journal of Molecular Biology 2003; 327(1):251-272.

[Att01] Attali D, Lachaud OJ. Delaunay conforming iso-surface, skeleton extraction and noise removal. Computational Geometry 2001; 19(2-3):175-189.

[Bat04] Battiato, S., Gallo, G., Messina, G. SVG rendering of real Images using data dependent triangulation. Proceedings of SCCG 2004 (2004) 185–192

[Béc02] Béchet E, Cuilliere JC, Trochu F. Generation of a finite element MESH from stereolithography (STL) files. Computer-Aided Design 2002; 34(1):1-17.

[Ber97] de Berg M, van Kreveld M, Overmars M, Schwarzkopf O. Computational Geometry. Algorithms and applications, Springer-Verlag Berlin Heidelberg, 1997.

[Cia97] Ciampalini, A., Cignoni, P., Montani, C., Scopigno, R. Multiresolution decimation based on global error. The Visual Computer, Vol. 13. Springer-Verlag (1997) 228–246

[Coo05] Cooper, O., Campbell, N., Gibson, D. Automatic augmentation and meshing of sparse 3D scene structure. Proceedings of 7th IEEE Workshop on Applications of Computer Vision, Breckenridge, USA, IEEE Computer Society (2005) 287–293

[Del34a] B. Delaunay. Sur la sphere vide. Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk 7 (1934), 793-800.

[Del34b] Б. Н. Делоне, А.Д. Александров, Н. Падуровым. Математические основы структурного анализа кристаллов, Москва, Матем. литература, 1934.

[Dem03] Demaret L., Iske A. Scattered data coding in digital image compression. In: Curve and Surface Fitting: Saint-Malo 2002, Nashboro Press, Brentwood, 2003, p. 107-117

[Dem04] Demaret, L., Dyn, N., Floater, M.S., Iske, A. Adaptive thinning for terrain modelling and image compression.Advances in Multiresolution for Geometric Modelling (2004) 321–340

[Dem06] Demaret L., Dyn N., Iske A. Image compression by linear splines over adaptive triangulations. Signal Processing 2006, 86(7):1604-1616

[Dev01] Devillers O, Pion S, Teillaud M. Walking in triangulation. In: Proceedings of 17th Annual Symposium on Computational Geometry, ACM, Medford, Massachusetts, USA, June 3-5, 2001. p. 106-114

[Dev98] Devillers O. Improved incremental randomized Delaunay triangulation. In: Proceedings of 14th Annual Symposium on Computational Geometry, ACM, 1998. p. 106-115.

[Dev99] Devillers O. On deletion in Delaunay triangulations. In: Proceedings of SCG'99, Miami Beach Florida, ACM, 1999. p. 181-188

[Dyn90] Dyn N, Levin D, Rippa S. Data dependent triangulations for piecewise linear interpolation. IMA Journal of Numerical Analysis 1990; 10: 127-154

[Ede92] Edelsbrunner H, Shah NR. Incremental topological flipping works for regular triangulations. In: Proceedings of 8th Annual Computational Geometry, 6/92, Berlin, Germany, ACM, 1992. p. 43-52

[Fac95] Facello MA. Implementation of randomized algorithm for Delaunay and regular triangulations in three dimensions. Computer Aided Geometric Design, Elsevier, 1995; 12:349–370.

[Flo76] Floyd RW, Steinberg L. An adaptive algorithm for spatial grey scale. In: Proceedings of the Society of Information Display (1976) 75–77

[Fur97] Furht B, Greenberg J,Westwater R. Motion Estimation Algorithms For Video Compression. Kluwer Academic Publishers, 1997, pp. 61-95

[Gal05] Galic, I., Weickert, J., Welk, M. Towards PDE-based image compression. In: Proceedings of VLSM 2005, Beijing, China, (2005) 37–48

[Gar99] García, M.A., Vintimilla, B.X., Sappa, A.D. Efficient approximation of grey-scale images through bounded error triangular meshes. In: Proceedings of IEEE International Conference on Image Processing, Kobe, Japan, IEEE Computer Society (1999) 168–170

[Gev97] Gevers, T., Smeulders, A.W. Combining region splitting and edge detection through guided Delaunay image subdivision. In: Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition (1997) 1021–1026

[God97] Godman JE, O'Rourke J. Handbook of discrete and computational geometry. CRC Press, 1997

[Gol97] Golias NA, Dutton RW. Delaunay triangulation and 3D adaptive mesh generation. Finite Elements in Analysis and Design 1997; 25(1997):331-341

[Gon02] Gonçalves G, Julien P, Riazanoff S, Cervelle B. Preserving cartographic quality in DTM interpolation from contour lines. ISPRS Journal of Photogrammetry and Remote Sensing 2002; 56(3):210-220.

[Gru05] Grundland, M., Gibbs, Ch., Dodgson, N.A. Stylized rendering for multiresolution image representation. In: Proceedings of SPIE, Vol. 5666, (2005) 280–292

[Gui85] Guibas LJ, Stolfi J. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. ACM Transactions on Graphics 1985; 4(2):75-123.

[Gui92] Guibas LJ, Knuth D.E, Sharir M. Randomized incremental construction of Delaunay and Voronoi diagrams. Algorithmica 1992; 7:381-413.

[Hil81] Hilbert D. Über die stetige Abbildung einer Linie auf ein Flächenstück. Mathematische Annalen 1981; 38 (1891): 459–460

[Hor81] Horn BKP, Schunck BG. Determining Optical Flow. Artificial Inteligence 1981; 17(1): 185–203

[Huf52] Huffman D. A. A method for the construction of minimum-redundancy codes. Proceedings of IRE 1952; 40(9): 1098-1101

[Jan08] Janák T. On interpolation for triangulation-represented digital image. In: Proceedings of CESCG 2008, Budmerice, Slovakia, pp. 51-58

[Jan09] Janák T. Interpolation methods for triangulation represented digital image. Bachelor thesis, University of West Bohemia, Czech Republic, 2009

[Kim99] Kim SS, Kim YS, Cho MG, Cho HG. A geometric compression algorithm for massive terrain data using Delaunay triangulation. In: Proceedings of WSCG, February 1999, Plzeň, Czech Republic, pp. 124-131

[Koh05] Kohout J. Delaunay triangulation in parallel and distributed environment, PhD Thesis, University of West Bohemia, Czech Republic, 2005

[Kre01] Kreylos, O., Hamann, B. On simulated annealing and the construction of linear spline approximations for scattered data. IEEE Transactions on Visualization and Computer Graphics, Vol. 7, IEEE Computer Society (2001) 17–31

[Luc81] Lucas BD, Kanade T. An Iterative Image Registration Technique with an Application to Stereo Vision. In: Proceedings of Imaging Understanding Workshop, 1981, pp. 121-130

[Mah07] Mahoney M. Data compression programs. December 2007. http://www.cs.fit.edu/~mmahoney/compression/

[Mar00] Marquant G. Reprèsentation par Maillage Adaptatif Dèformable pour la Manipulation et la Communication d'Objets Vidèo. Universit de Rennes, 2000.

[Mar80] Marr D, Hildreth EC. Theory of edge detection. In: Proceedings of the Royal Society, Vol. 207, London, 1980, p. 187–217

[MSDN09] Microsoft Corporation. Color. Microsoft Developer Network; July 2009. http://msdn.microsoft.com/en-us/library/aa511283.aspx

[Mue97] Mueller C. Hierarchical graphics databases in sort-first. In: Proceedings of IEEE Symposium on Parallel Rendering, 1997. p. 49-57

[Mul03] Mulchrone KF. Application of Delaunay triangulation to the nearest neighbour method of strain analysis. Journal of Structural Geology 2003; 25(5):689-702.

[Nik09] Nikon. Creation and Light; July 2009. http://www.nikon.com/about/feelnikon/light/chap03/sec01.htm

[Nis01] Nishioka T, Tokudome H, Kinoshita M. Dynamic fracture-path prediction in impact fracture phenomena using moving finite element method based on Delaunay automatic mesh generation. International Journal of Solids and Structures 2001; 38(30-31):5273-5301.

[Noc05] Nock R, Nielsen F. Semi-supervised statistical region refinement for color image segmentation. Pattern Recognition, Elsevier, 38 (2005):835 – 846

[Oka92] Okabe A, Boots B, Sugihara K. Spatial tesselations: Concepts and applications of Voronoi diagrams. John Wiley & Sons Ltd, 1992.

[Oku96] Okusanya T, Peraire J. Parallel unstructured mesh generation, Presented at 5th Int. Conf. on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields, Mississippi, 1996

[Oku97] Okusanya T, Peraire J. 3D Parallel unstructured mesh generation, http://citeseer.nj.nec.com/article/okusanya97parallel.html

[Ost99] Ostromoukhov V, Hersch RD. Stochastic clustered-dot dithering. Color Imaging: Device-independent Color, Color Hardcopy, and Graphic Arts IV, SPIE 1999; 3648:496 – 505.

[Par03] Park JH, Park HW. Fast view interpolation of stereo images using image gradient and disparity triangulation. Signal Processing: Image Communication 2003; 18(5):401-416.

[Part03] Partyk M., Polec J., Kolingerová I. Hybrid scheme with triangulations for transform coding. Radioengineering, 12(3), September 2003

[Pra00] Prasad L, Rao L.R. A geometric transform for shape feature extraction. In: Proceedings of the 45th SPIE Annual Meeting, San Diego,CA, 2000.

[Pra06] Prasad, L., Skourikhine, A.N. Vectorized image segmentation via trixel agglomeration. Pattern Recognition, Vol. 39, Elsevier (2006) 501–514

[Pun08] Puncman P. Použití triangulací pro reprezentaci videa. Master thesis, University of West Bohemia, Czech Republic, 2008.

[Rad99] Radke J, Flodmark A. The use of spatial decomposition for constructing street centerlines. Geographic Information Services 1999; 5(1):15-23.

[Ril98] Rila L. Image coding using irregular subsampling and Delaunay triangulation. In: Proceedings of SIBGRAPI, 1998, p. 167–173

[Ros99] Rosignac J. Edgebreaker: connectivity compression for triangle meshes. IEEE Transactions of Visualization and Computer Graphics 1999; 5(1999): 47–61

[Rul09] Rulf M. Využití triangulizací pro reprezentaci digitalizovaného obrazu. Bachelor thesis, University of West Bohemia, Czech Republic, 2009.

[Sch99] Schewchuk JR. Lectures notes on Delaunay mesh generation. Department of Electrical Engineering and Computer Science, University of California at Berkley, CA 94720, 1999.

[Slo92] Sloan SW. A fast algorithm for generating constrained Delaunay triangulations. Computers & Structures 1992; 47(3):441-450

[Sou07a] SourceForge contributors. PeaZip – free archiver utility. SourceForge.net; December 2007. http://peazip.sourceforge.net/

[Sou07b] SourceForge contributors. QUAD. SourceForge.net; December 2007. http://quad.sourceforge.net/

[Su04] Su D., Willis P. Image interpolation by pixel level data-dependent triangulation. Computer Graphics Forum, Vol. 23 (2004) 189–201.

[Syk08] Sýkora R. Komprese barevných digitálních obrazů s využitím triangulace. Bachelor thesis, University of West Bohemia, Czech Republic, 2008.

[Tek00] Tekalp AM, Ostermann J. Face and 2D mesh animation in MPEG4. Signal Processing: Image Communication 2000; 15(4-5):387-421.

[Uhl05] Uhlíř K, Skala V: Reconstruction of damaged images using radial basis functions. In: Proceedings of EUSIPCO 2005, Istanbul, Turkey, 2005, p. 160.

[Usc07] University of Southern California. The USC-SIPI Image Database. Signal & Image Processing Institution, University of Southern California, ed. Allan Weber; December 2007. http://sipi.usc.edu/database/

[Var07] Varga M. Využití tetrahedralizace jako alternativy k objemovým datům. Master thesis, Charles University, Czech Republic, 2007

[Vas07] Váša L., Skala V. CoDDyAC: Connectivity driven dynamic mesh compression. In: Proceedings of IEEE 3DTV Conference, Kos, Greece, May 7-9, 2007.

[Vig97] Vigo M. An improved incremental algorithm for construting restricted Delaunay triangulations. Computers & Graphics 1997; 22:215-223.

[Vig00] Vigo M, Pla N. Computing directional constrained Delaunay triangulations. Computer & Graphics 2000; 24:181-190.

[Vom08] Vomáčka T. Delaunayova triangulace pohybujících se bodů v rovině. Master thesis, University of West Bohemia, Czech Republic, 2008

[Vom09] Vomáčka T, Puncman P. A novel video compression scheme based on kinetic Delaunay triangulation. In: Proceedings of Algoritmy 2009, Podbanske, Slovakia, March 2009, pp. 372-381

[Wal00] Walkington NJ, Antaki JF, Blelloch GE, Ghattas O, Melcevic I, Miller GL. A parallel dynamic-mesh Lagrangian method for simulation of flows with dynamic interfaces. In: Proceedings of the 2000 ACM/IEEE conference on Supercomputing (CDROM), November 2000, Dallas, Texas, United States. p.26

[Wat81] Watson DF. Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes, Computer Journal 1981, 24(2):167-172.

[Wei98] Weimer H, Warren J, Troutner J, Wiggins W, Shrout J. Efficient Co-Triangulation of Large Data Sets. IEEE Visualization 98, 1998, p. 119 – 126

[Wik07a] Wikipedia contributors. 7z. Wikipedia, The Free Encyclopedia; December 2007. http://en.wikipedia.org/7z

[Wik07b] Wikipedia contributors. Bzip2. Wikipedia, The Free Encyclopedia; December 2007. http://en.wikipedia.org/wiki/bzip2

[Wik07c] Wikipedia contributors. PAQ. Wikipedia, The Free Encyclopedia; December 2007. http://en.wikipedia.org/wiki/PAQ

[Wik07d] Wikipedia contributors. Distance. Wikipedia, The Free Encyclopedia; December 2007. http://en.wikipedia.org/wiki/Distance

[Wik07e] Wikipedia contributors. Principal components analysis. Wikipedia, The Free Encyclopedia; December 2007.
http://en.wikipedia.org/wiki/Principal_components_analysis

[Wik07f] Wikipedia contributors. Run-length encoding. Wikipedia, The Free Encyclopedia; December 2007. http://en.wikipedia.org/wiki/Run-length_encoding

[Wik08a] Wikipedia contributors. Run-length encoding. Wikipedia, The Free Encyclopedia; March 2008. http://en.wikipedia.org/wiki/Phong_shading

[Xia02] Xiao Y, Yan H. Text region extraction in a document image based on the Delaunay tessellation. Pattern Recognition 2003, 36(3):799-809

[Yu01] Yu, X., Morse, B.S., Sederberg, T.W. Image reconstruction using data-dependent triangulation. IEEE Computer Graphics and Applications, Vol. 21, IEEE Computer Society (2001) 62–68

[Žal03] Žalik B, Kolingerová I. An incremental construction algorithm for Delaunay triangulation using the nearest-point paradigm. International Journal of Geographical Information Science 2003, 17(2):119-138.

[Žar98] Žára J, Beneš B, Felkel P. Moderní počítačová grafika. Computer Press, 1998, ISBN 80-7226-049-9