TRIANGULATIONS AND THEIR APPLICATION IN SURFACE RECONSTRUCTION

Publisher:
Print Partner IPSKAMP,
Capitool 25 (Business & Science Park),
Postbus 333, 7500 AH Enschede.
www.ppi.nl

The research of the author was made possible by the NWO (STW)
(Dutch Organisation for Scientific Research), project No. TWI4816.

ISBN 90-365-2080-0

TRIANGULATIONS AND THEIR APPLICATION IN SURFACE RECONSTRUCTION

DISSERTATION

to obtain
the doctor's degree at the University of Twente,
on the authority of the rector magnificus,
prof. dr. F.A. van Vught,
on account of the decision of the graduation committee,
to be publicly defended
on Friday 24 September 2004 at 13.15

by

Alexandre Igorevitch Netchaev
born on 22 November 1974
in Topki, Russian Federation, the USSR

Dit proefschrift is goedgekeurd door de promotor
Prof. dr. C.R. Traas

de assistent-promotoren
Dr. L. Alboul
Dr. R.M.J. van Damme

to my parents
Vera and Alexandre

Contents

Chapter 1

Introduction

The construction of a triangulation of a scattered data set in $\mathbb{R}^3$ is an important issue in surface reconstruction and widely used in many applications, such as computer graphics and vision, mesh generation, terrain modelling, tomography, reverse engineering, pattern recognition, computational geometry, cartography, geology, stereology, architecture, medical imaging and many other related fields. For example, in computer graphics and vision the triangulation is a framework for drawing and visualization of an object. In mesh generation the triangulation of the domain is used for solving partial differential equations [BE95, Geb99, Sch02]. In terrain modelling the triangulation is a part of the earth surface [AdGB00, Bou94, GD02]. In tomography it is a surface of the investigated organ [HAP, TSC+01]. In reverse engineering it is a three-dimensional (**3D**) model of an object [Cha94, CW99, Flo96, HAP, RJPC02]. In general, in all fields, where one needs to recover the surface of a solid object from **3D** scattered data, it is necessary to construct an initial triangulation. An initial triangulation is constructed from a given data set of points with seemingly no structure. These points can be measured directly on the boundary of an object using **3D** sensors or laser range scanning systems. They can also be taken from the segmentation of a **3D** volumetric image, contact probe digitizers, radar and seismic surveys.

In this work we deal with the problem of surface reconstruction. Under surface reconstruction one understands finding a surface which approximates a set of given points in some way and is the "best" (a good) representation of the original surface. A widely used approach is to describe the resulting surface by a triangulation, *i.e.,* by a triangulated polyhedral surface that spans the given three-dimensional data. In general, the given sample data points are the vertices of a triangulation. Starting from the data points, of which we know only their positions, we are interested in constructing an initial triangulation induced

by the proximity relationship of the points on the surface of the object. This latter task seems ill-defined. Indeed, given the data, one can construct many polyhedral surfaces that span the data, each of them might be a good representation of some underlying surface. However, if we single out an underlying surface; then most of the triangulations will be quite unsatisfactory. On the other hand, if the data are originated from an unknown surface, the proximity relationships are also not known. Therefore, we reformulate our goal, and say that we want to construct an initial triangulation that interprets the data in an optimal way. In case that the underlying surface is known, our initial optimal triangulation will be an approximation to the surface, *i.e.*, from the triangulation we will be able to determine (within a certain range) *metric* properties of a surface, retrieve its *shape*, and/or, as it is often needed in many applications, just to get a *visualized model* of the surface. (Here we presume, that a triangulation corresponds to topological characteristics of the surface, *i.e.*, possesses the same genus and connectivity). The initial triangulation might then be used for a further elaboration of the surface, for example such as its smoothing by a subdivision scheme. On the other hand, if the data are taken from an irregular (non–smooth) surface, a triangulation (a polyhedral surface) might be the only "reliable" approximation of this surface.

The problem of finding the "best" initial triangulation can be stated now as an *optimization problem*, we want to determine the "best" triangulation not in some abstract way, but with respect to certain optimality criteria.

Our ultimate research goal is to study triangulations of discrete data with respect to their discrete curvatures (total absolute Gaussian curvature, mean curvature). Why curvatures? As we said above, we want to obtain an initial triangulation that would reflect the shape of the underlying surface; and the curvatures truly describe the shape of surfaces.

We want to investigate which properties the optimal triangulations possess, for instance, triangulations on which the corresponding curvatures reach their minimum.

Concepts of discrete curvatures are of growing interest for geometric modelling [AER04, AvD96, Bob04, DMSB02, Gar04, MD02]. We can single out several reasons for this. Beside the attempts to reduce high computational expenses, one of the reasons is that many applications deal with three-dimensional discrete point data, and therefore, only a discrete approach makes sense. Another reason is that a polyhedral model of the underlying object, which is represented by discrete data, is the simplest way to obtain a preliminary sketch of the given object. In general, such a model is given in the form of a triangulated polyhedral surface, or simply a triangulation. Polyhedral surfaces belong to so-called non–regular surfaces, and for non-regular surfaces concepts of curvatures similar to classical curvatures are well determined [AvD96, AZ67, BK79]. Let us note that

the current progress in the field of discrete differential geometry is to a large extent due to its relevance for computer graphics and visualization.
Our research with respect to discrete curvatures is limited to their use as optimality criteria. Motivation, for example, to use the criterion of minimizing the total absolute Gaussian curvature is related to the concept of Tight submanifolds [BK97, Kui70]. One of the main properties of two–dimensional tight submanifolds in $\mathbb{R}^3$ is that they possess the minimal total absolute curvature, or, equivalently, the height function on a tight surface will have the minimum number of non-degenerate critical points that a height function can have on a closed surface. Therefore, a tight surface of genus 0 is a convex surface. We deal with the data taken from some underlying surface, which is in general non-convex. However, we can assume that the data are taken in such a way, that all important features (creases, curvatures, etc.) of a surface can be extracted from the data, and that the representation of a surface as a triangular mesh does not add features not present in the data. Therefore, the properties of Tight submanifolds make the idea to construct a triangulation of minimum total absolute curvature very appealing. Indeed, surface triangulations of the minimum corresponding analogue of this curvature (*i.e.,* total absolute extrinsic curvature, or **MTAEC**), were introduced some time ago [AvD95a, AvD95b]. The MTAEC triangulation coincides with the convex triangulation of the data if the data are in convex position, but the properties of triangulations with MTAEC for non-convex data are still mostly unknown (see [AvD96] for a review). Triangulations of minimum mean curvature were also considered [AKTvD99, DHKL01], and experimental results seem very promising, but there is also very little known about their properties.
To tackle this problem the idea was to generate all possible admissible triangulations (*i.e.,* without self–intersections) of genus 0 on small data sets (for example up to 15 data points) and compute their discrete curvatures, finding out the optimal ones and establish their properties. To attain this objective we distinguish two approaches in our work:

1. Combinatorial approach;
2. Geometrical approach.

In this work we consider both of them. Therefore, for clarity, we divide our work into two parts. In part I we consider combinatorial triangulations of a sphere using notions from combinatorial theory, theory of groups and topological graph theory; in part II we consider triangulations reconstructed from real data, *i.e.,* polyhedral triangulated surfaces that span the given data. Let us note that we work with closed surfaces; therefore triangulations represent a polyhedral closed surface of a certain topological type: we concentrate on triangulations that are topologically equivalent to the **2D** sphere. These triangulations can

also be viewed as geometrical realizations of combinatorial triangulations on the sphere.

Generating and enumerating a specific class of objects are fundamental problems in discrete mathematics, graph theory, computational geometry, the cell growth problem and many other fields. In part I we concentrate on the problem of combinatorially generating triangulations of the topological type of the sphere, which can be represented by planar graphs. We consider the construction of such triangulations without fixing the coordinates of points and develop different algorithms for generating all possible and all non-isomorphic triangulations for a given number of points. There are many publications dedicated to this problem [AK96, Avi96, BF67, BM01, BP71, BP72, BP74, Har60, Har68, MAT, Mul65]. We single out the works of R. Bowen and S. Fisk [BF67], L.W. Beineke and R.E. Pippert [BP74], G. Brinkmann and B. McKay [BM01] because they are directly related to our research.

Triangulations, constructed over discrete data, can be viewed as labelled triangulations. The number of them is growing very fast with respect to the number of vertices, therefore explicit generation of all labelled triangulations is possible only for a small number of vertices. Therefore, we came to the problem of generating all possible unlabelled triangulations, and consequently, to the problem of generating all non-isomorphic ones. The latter problem leads to the problem of generating non-isomorphic triangulations without checking on isomorphisms. This problem has partially been solved: we are able to generate specific subclasses of triangulations without any control on isomorphisms.

We tried to explore various approaches, some of these approaches were new, and some other ones inevitably turned out to be similar to the approaches known in literature. We give the corresponding references whenever it is appropriate.

Part I, besides being a report on our original research, can also be used as a survey of various combinatorial methods to construct triangulations.

In part II we concentrate on the problem of the geometrical construction of triangulations, by using data sets of points with fixed coordinates, and study the properties of total absolute curvature and mean curvature. We want to develop algorithms for constructing a "good" initial triangulation from a given data set and then to find an optimal one by edge flipping algorithms with respect to different optimality criteria. To attain the objective we consider the combinatorial triangulations, which were constructed in part I, and fix their vertices by assigning them the coordinates of the points of the given data set. As we will see in part I, the difficulty of this way of construction is that the number of combinatorial possibilities is very high, even for small data sets, and to treat them all is still an impossible computational task.

Therefore, we reconsider our initial idea to compare discrete curvatures of all possible admissible triangulations on small data sets; and decide to address the

problem from a somewhat different point of view: given a data set of points with fixed coordinates; we construct an initial triangulation and only then we study discrete curvature criteria on it. Starting from this initial triangulation, we proceed with construction of an optimal triangulation with respect to a chosen criterion. One simple way for constructing an initial triangulation is to add points recursively one by one as we do in part I. But the geometric problems we are faced with are:

- in which sequence to add new points
- how to avoid self–intersections in the surface;
- is there a difference in the construction strategy for objects of various genera;
- what to do if an object is not closed.

Well-known algorithms for generating an initial triangulation are based on using the convex hull, Delaunay triangulations or Voronoi diagrams [AB99, AB02, ABK98, AC99, BG92]. But a robust algorithm which generates a triangulation with only the knowledge of coordinates of points seems not to exist.
For the sake of argument, suppose an initial triangulation for a scattered data set of points is constructed by one or another method. This triangulation is not always good for every application. In most articles authors consider, therefore, the problem of starting from an initial triangulation to investigate different methods for finding an optimal one [AvD95b, BE95, DHKL01, DLR90, MD02]. Optimization usually consists of transforming an initial triangulation via a sequence of transformations to some final triangulation, which is better with respect to the given criterion. The operation of transformation should preferably be simple as well as general enough in order to be able to reach the optimal triangulation from any initial one. The most natural transformation is the geometrical flip, or simply, the edge flip. The edge flip algorithm (EFA) was proposed by Charles Lawson in 1972 [Law72] for triangulations in the plane. It has also been shown that for any two triangulations T_1 and T_2 of a given planar point set V, there is always a sequence of Lawson's operations transforming T_1 into T_2. However, in space, the EFA produces self-intersections [AHA02]. Nevertheless, it is still possible to use it by observing certain requirements [Alb03, AvD02]. Using EFA and applying an optimality criterion, an initial triangulation can be transferred into a different one up to the point that no edge flipping is possible any more. Such a triangulation will be locally optimal with respect to the optimality criterion. There are many criteria of optimization, such as:

- minimization of the area of the resulting object [O'R87];

- minimization of the (total) weight of a triangulation, where under the total weight one mean the sum of the length (cost) of all the edges of a triangulation [Epp92, FNP96, Tót04];
- heuristic criteria, based on minimizing a measure of roughness of the resulting object. One such measure is the jump in normal derivatives (JND) and the other measure is the angle between normals (ABN) [DLR90];
- methods based on minimizing a certain functional, like the energy of a bending plate, constrained by the interpolation conditions [QS90];
- methods based on minimizing discrete analogues of the integral absolute Gaussian curvature and integral absolute mean curvature, and related energies, such as minimization of the Willmore energy (integral over the squared Mean curvature) [AvD95a, AvD95b, AvD96, Bob04, DHKL01, MD02];

For a data set with many points it is very hard to presume that one can find the *global* optimum. In general, any local minimum of energy is considered as a solution to the optimization problem, mostly because the global minimum is hard to reach. However, the global minimum might also be not unique, moreover, it might be far away from the input configuration [FMS03]. A local minimum, which is as close as possible to the initial configuration, might be more meaningful. Therefore, in part II of the thesis we want to consider methods for constructing an *initial* triangulation using the following approach: our goal is to find an algorithm which can construct a good (not necessarily the best) triangulation based on a certain criterion. Moreover, we wish to study the behavior of certain criteria (such as based on curvatures), trying to answer the question which criterion is the best.

Other research in Surface Reconstruction

In recent years, the problem of surface reconstruction in three-dimensions has received much attention from researchers in *computer graphics* as well as from researchers in *computational geometry*. The algorithms which are used in computer graphics, typically compute an approximating surface, that is, a surface passing close by, rather than exactly through, the original sample points. The algorithms devised by computational geometers typically restrict attention to surfaces on the original sample points, usually starting with a Delaunay triangulation on a carefully chosen subset of the sample points [AB99].
The first widely known reconstruction algorithm in computer graphics was done by Hoppe et al. [HDD+92] in 1992. They presented an algorithm which reconstructs a surface in three dimensional space with or without boundary from a

set of unorganized points scattered on or near the surface as the zero set of a signed distance function. The algorithm is based on the idea of first computing a signed distance function f from the data points, such that f estimates the signed geometric distance to the unknown surface. Then the zero set of f is contoured by a piecewise-linear surface using the marching cubes algorithm. To define the signed distance from the unknown surface, Hoppe et al. compute a best-fit tangent plane for each data point, and then find a coherent orientation for the surface by propagating the normal direction from point to point, using a precomputed minimum spanning tree to favor propagation across points whose associated normals are nearly parallel.

Curless and Levoy [CL96] use a similar algorithm for data samples collected by a laser range scanner, from which they derive tangent plane information. This algorithm sums anisotropically weighted contributions from the samples to compute a signed distance function, which is then discretized on voxels to eliminate the marching cubes step.

These two computer graphics algorithms are quite successful in practice, but have no provable guarantees. Indeed there exist arbitrary dense sets of samples for which the algorithm of Hoppe et al. fails [AB99].

In computational geometry almost all reconstruction algorithms are based on the Delaunay complex of the sample points. In three dimensions the Delaunay complex is a tetrahedralization of the point set. It is well studied and has found many applications over the years. The first Delaunay based reconstruction algorithm was given by Boissonnat [Boi84]. He proposed two approaches for constructing a triangulation from a given data set of points. The first approach is "local" and surface-based, whereas the second one is "global" and volume-based.

The first approach uses some results from differential geometry and takes advantage of the fact that a surface in $\mathbb{R}^3$ is essentially two–dimensional. It is based on the idea that one starts with creating an edge between the two closest points. Then one chooses a third point and adds it to the constructed edge such that they form a triangle. The other points are added successively to an edge of the current triangulated boundary (new triangles are created), until all points have been included. Theoretical results guarantee the quality of the approximation. But the approach can only be applied if the discretization is fine enough.

The second approach takes the discrete nature of the problem into account and uses a global data structure, famous in the field of computational geometry: the Delaunay triangulation. It is based on the idea of first computing the **3D** Delaunay triangulation of the convex hull of the set of points, and then sculpturing this convex hull by removing the exterior tetrahedra incrementally, starting from outside, until all points are on the boundary of the shape, or no tetrahedra can be further removed. A drawback of such an approach is that

no change of the topology is allowed and consequently, it is impossible to get a surface formed of several connected components or having holes. The piecewise-linear model generated by this approach contains volumetric information that is useful for other applications, but this approach is more costly in the worst case.

Another Delaunay type reconstruction algorithm is based on using the alpha-complex which is a subcomplex of the Delaunay complex and usually is computed via the Delaunay complex. Edelsbrunner and Mücker in [EM94] generalized the notion of a convex hull to create surfaces called alpha-shapes. Alpha-shapes represent a finite set of points at different levels of detail. The major shortcoming of this approach is the determination of the parameter α, the radius of the cutting sphere. The main difficulty of the algorithm is that the best reconstruction may require different α in different places and the choice of α has to be determined experimentally. Bernardini and Bajaj developed several automatic reconstruction methods based on alpha-shapes and algebraic-patch fitting [BBX95]. Later, in [BB97] they gave a formal characterization of the reconstruction problem and proved that, the reconstructed alpha-shape is homeomorphic to the original object and approximates it within a fixed error bound if certain sampling requirement are satisfied. For improving the reconstruction results the next authors amended the alpha-shapes algorithm in this a way: Guo et al. [GMW97] used visibility algorithms, Teichmann and Capps [TC98] used density scaling and anisotropic shaping, Xu and Harada [XH03] scaled the α ball according to the point's density.

For the two–dimensional case, Attali [Att98] introduces normalized meshes to give bounds on the sampling density within which the topology of the original curve is preserved. His method is proved to provide the exact solution in **2D** but unfortunately the result could not be extended to **3D**. Amenta et al. in [ABE98] introduced the notion "crust" for curve reconstruction and gave a guarantee for this method. Then they made an extension of "crust" to three-dimensions, which they called "Voronoi filtering" [AB99]. The idea of the algorithm is based on the assumption that the distance between samples is proportional to the distance to the "medial axis". They used a Voronoi filtering approach, based on **3D** Voronoi diagrams and Delaunay triangulations, to construct an interpolating shape of the sample points called "crust". It was the first algorithm with a theoretical guarantee. For sufficiently dense sampling their algorithm gives a piecewise-linear surface which is homeomorphic and geometrically close to the original surface. However, for non-smooth surfaces Amenta and Bern were not able to give any guarantee. In fact their algorithm has difficulties in reconstructing non-smooth or badly sampled parts of a surface properly. Later, in [ACDL00], Amenta et al. presented a simplified version of the reconstruction algorithm, which holds the same theoretical guarantee. Recently, other results

in surface reconstruction appear but almost all of them are only improvements of already existing methods [AGJ02, LCK03, PB01].

Overview of the thesis

Below we briefly describe what has been done in each section of this thesis. In chapter 2 we give general definitions, theorems and some basic notions and concepts for introducing the reader into the basic knowledge of Graph Theory. In chapter 3 we consider two recursive methods for generating all possible triangulations. Both algorithms need to check on isomorphisms. We only want to generate triangulations that are really different. An algorithm that generates one particular triangulation more than once is not efficient. Firstly, in section 3.1, we estimate an upper bound for the number of all possible triangulations, which grows very fast as a function of the size of the data sets. We also show an exponential approximation to the number of all possible triangulations, which is greater than N^{3N}. Next, in section 3.2, we present a recursive method for generating triangulations of $(N+1)$ points from triangulations of N points by adding a new vertex to three points of the same face, to four points of two adjacent faces and so on, as long as it is possible. The method generates all triangulations for N points and the number of them is too large (see an upper bound above). That is why we decided to use the concept of isomorphisms of graphs, corresponding to triangulations. In section 3.2.1 we present the method for generating all possible non–isomorphic triangulations for N points which needs checking on isomorphisms between all triangulations. However, the operation of checking on isomorphisms turns out to be computationally too expensive. Therefore, the next step is to try to avoid checking on isomorphisms as much as possible. We introduce a technique of indexing vertices (see section 3.2.2) which uses the concept of automorphisms of graphs (triangulations in our case) and allows us to reduce the number of tests on isomorphisms between two triangulations.
Subsequently we constructed another algorithm for generating all possible non–isomorphic triangulations, based on the idea that a triangulation can be divided by cross-sections through one edge into a few simple polyhedral blocks, namely k-pyramids and wedges. We presumed that two different cross–sections have only one edge in common. However, it still relies on checking on isomorphisms. We call the algorithm the "method of cross-sections" (see section 3.3). The great advantage is, however, that this algorithm needs much less checking on isomorphisms than the previous method. Our proposition to separate all triangulations by cross–sections that share at most one edge turns out to be wrong for generating a triangulation which is close to a six–regular one. Therefore we continued our search for a better algorithm for generating all possible non–

isomorphic triangulations without using any checking on isomorphisms.
As a result, we found an algorithm for generating all non-isomorphic dissectible triangulations, which is described in chapter 4. We divide all triangulations into two subsets. One subset consists of all triangulations which have at least one vertex of degree three and the other subset contains all triangulations which have no vertex of degree three. In section 4.1 and section 4.2 we describe a method for generating triangulations of the first subset without any checking on isomorphisms. The idea behind this method is to add simultaneously l new points in a special way, where $k \leq l \leq 2(N-2)$ and k is the number of vertices of degree three.
In section 4.3 we present a method for generating triangulations of the second subset of triangulations, that is, triangulations without vertices of degree three. Again, we divide this subset of triangulations into two types. One type consists of all triangulations which are constructed by gluing together a number of polyhedra through one face and the other type contains the remaining triangulations. An algorithm for generating the first type of triangulations is relatively simple and leads to solving a combinatorial problem. An idea of an algorithm for generating the second type of triangulations is to define for every triangulations l "domains"[1] with k_i vertices of degree four inside ($i = 1..l$), then to divide each domain in m_i subdomains in a special way and to add at least l and at most $\sum_{i=1}^{l} m_i$ new points of degree four into the subdomains of a triangulation. The method generates non-isomorphic triangulations, but some of them are missed in this construction.
In chapter 5 we consider the two–dimensional case of dissectible triangulations – dissectible polygons. For generating them, vertices of degree two are taken as vertices for adding new points to edges (faces in **3D**) of a polygon. In section 5.1 one can find the main algorithm for **2D**, and results. Using the property of automorphisms we make two approaches of the main algorithm for dissectible polygons (see section 5.2 and section 5.3). As a result, both approaches require adding $\tilde{l}$ new points per time, where $2 \leq \tilde{l} \leq 2k$. This number is much lower than in the main algorithm, discussed in section 5.1. The proofs and algorithms for generating all non–isomorphic dissectible polygons by the first and the second approaches can be found in section 5.2 and section 5.3. Next we adapt the algorithm for dissectible polygons to the combinatorial problem of generating triangular animals[2]. By imposing some restrictions on that algorithm it is possible to construct all tree-like and pseudo–connected triangular animals. This all is described in chapter 6.
In chapter 7 we give the reader the general definitions, theorems and some basic

[1] See definition 4.3.1 of domain.
[2] An animal is the collection of cells of equal shape starting from a single one, which grows step by step in the plane by adding at each step a cell in such a way that the new cell has only connection with a side of an already presented cell.

notions which are used in part II. In particular, in section 7.1 one can find basic notions on differential geometry, namely, definitions of regular curves and surfaces, principal curvatures, mean and Gaussian curvatures. In section 7.2 the discrete analogues of mean and Gaussian curvatures are briefly described following the definition which was given by L. Alboul and R. van Damme in 1995 [AvD95a, AvD96]. Then in section 7.3 and 7.4 one can find a brief description of different optimization criteria for generating an optimal triangulation.

In chapter 8 we present a multi edge flipping algorithm and the method for generating an initial and optimal triangulation. Namely, in section 8.1 we describe the edge flipping algorithm (EFA) of Charles Lawson [Law72]. By this algorithm any triangulation can be transferred into any other one; thus, we can find all possible geometrical triangulations for the given data set. To find the optimal one, one applies an optimality criterion, but in that case we show that EFA is not sufficient for finding an optimal triangulation if an initial one is very "bad". Because there are situations when the energy increases after swapping one edge, but it decreases after swapping two or more edges. Therefore, we introduce the multi edge flipping algorithm (MEFA) which in combination with the same optimality criterion gets closer to optimal triangulation than EFA for the same initial one. There is only one problem with MEFA: How many edges are necessary to swap? The answer to this question is very difficult to find because it depends on the "badness" ("poorness") of the initial triangulation. In section 8.2 we describe a method for constructing an initial triangulation for a given data set which uses as optimality criterion the minimization of the absolute mean curvature. In particular, in subsection 8.2.1 we consider a recursive method for constructing an initial triangulation (the theoretical aspect of it was described in section 3.2). Because in part II we deal with geometric problems, we introduce the concept of visibility between a point and a surface. Then, using visibility, we directly apply the recursive method for constructing an initial triangulation and after each recursion step we apply an optimality criterion. The resulting triangulation appears to be very "bad" because of the non-organized data set. Therefore in subsection 8.2.2 we introduce a way of reordering a given data set before starting the construction of an initial triangulation. The result is a better triangulation than before. But applying an optimality criterion in every step of the recursion is not efficient. Therefore, in subsection 8.2.3 we show that the optimality criterion can be applied after adding k new points to a triangulation per time ($1 \leq k \leq N$). Then we present a construction algorithm, which is divided into three steps: reordering the data set, constructing the "carcase"[3] and adding the remaining points - and give four different ways for adding a new point. In section 8.3 we give a brief summary of chapter 8 and describe weak

[3] A "carcase" is a triangulation which is constructed by adding new points in a special way, described in section 8.2.3.

and strong points of the algorithm. In the appendix one can find the pictures of the constructed initial triangulations in the main steps of the algorithm.
In chapter 9 we give suggestions for further possible work with triangulations and describe problems which are not studied yet.

Part I

Generating triangulations

Chapter 2

General notes

2.1 Basic notions of Graph Theory[1]

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a mathematical structure consisting of two sets $\mathcal{V}$ and $\mathcal{E}$. The elements of $\mathcal{V}$ are called *vertices* and the elements of $\mathcal{E}$ are called *edges*. With N (or V) and E we denote the number of all vertices and edges, respectively, of a graph $\mathcal{G}$, $N = \mathcal{V}$ and $E = \mathcal{E}$. For example, the objects in figure 2.1 are graphs.

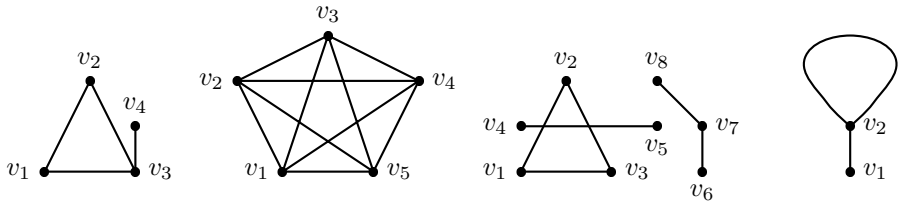


Figure 2.1: Examples of graphs

The graph with only one vertex is called *trivial* graph, all other graphs are *nontrivial*. A graph is *simple* if it has no loops (vertices joined to themselves, as in figure 2.1(d)) and the same pair of vertices is not joined by more than one edge. A graph in which each pair of distinct vertices is joined by an edge is called a *complete* graph, an example of such a graph is the graph of figure 2.1(b). Below we give some general definitions concerning graphs and to gain a better understanding of them we present some examples [BM76, Tru94]:

[1] Most of the definitions, theorems and examples of this section were taken from the book of R.J. Trudeau "Introduction to graph theory" and the book of J.A. Bondy and U.S.R. Murty "Graph theory with applications". We refer the reader to the original text for more details.

Definition 2.1.1 *The degree of a vertex $d(v)$ is the number of edges incident with this vertex.*

Theorem 2.1.2 *The sum of the degrees of all vertices of a graph $\mathcal{G}$ is equal to the double number of edges of $\mathcal{G}$*

$$\sum_{v \in V} d(v) = 2E$$

Definition 2.1.3 *A walk in a graph is a sequence $v_1 v_2 v_3 ... v_k$ of not necessarily distinct vertices v_i in which v_1 is joined by an edge to v_2, v_2 is joined by an edge to v_3, ..., and v_{k-1} is joined by an edge to v_k. The walk $v_1 v_2 v_3 ... v_k$ is said to join v_1 and v_k. The walk $v_1 v_2 v_3 ... v_k$ is closed if v_1 and v_k are the same vertex; otherwise the walk is open.*

Definition 2.1.4 *A walk for which the edges $e_i \in E$ are pairwise distinct is called a trail, a closed walk with that property is a closed trail. A closed trail with $N > 2$, for which the v_j are pairwise distinct (except $v_0 = v_N$) is called a cycle. An empty cycle is a cycle without any points inside.*

Definition 2.1.5 *A graph is said to be connected if every pair of vertices is joined by a walk. Otherwise a graph is said to be disconnected.*

Definition 2.1.6 *The connectivity $\kappa(\mathcal{G})$ of a graph $\mathcal{G}$ is the minimum number of vertices whose removal results in a disconnected or trivial graph.*

Definition 2.1.7 *A graph $\mathcal{G}$ is n–connected if $\kappa(\mathcal{G}) \geq n$.*

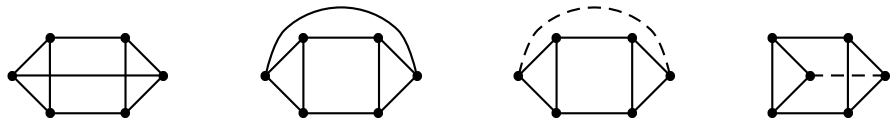
The graphs of figure 2.1(a) and figure 2.1(b) contain a *walk* for every pair of vertices and the graph of figure 2.1(c) does not. Indeed, $\{v_1, v_2, v_3, v_4\}$ is a *walk* and a *trail* for the graph of figure 2.1(a), $\{v_1, v_2, v_4, v_5, v_2, v_3, v_1\}$ is a *closed walk* and $\{v_1, v_2, v_3, v_4, v_5, v_1\}$ is an *empty cycle* (a *closed trail*) for the graph of figure 2.1(b). The graphs of figure 2.1(a) and figure 2.1(b) are called *connected* and the graph of figure 2.1(c) is called *disconnected* (there is no connection between sets of vertices $\{v_1, v_2, v_3\}$, $\{v_4, v_5\}$ and $\{v_6, v_7, v_8\}$). Moreover, deleting vertex v_3 of the graph of figure 2.1(a) gives a disconnected graph. It means that its connectivity is $\kappa = 1$. The graph of figure 2.1(b) can not be disconnected by removing any number of vertices, but after removing four vertices it becomes the trivial graph, therefore, its connectivity is $\kappa = 4$.

Definition 2.1.8 *A graph is planar if it can be drawn in a plane without edge–crossings.*

Definition 2.1.9 *A graph $\tilde{\mathcal{G}}$ is called a planar embedding of $\mathcal{G}$ if $\mathcal{G}$ is redrawn in the plane without edge–crossings.*

Definition 2.1.10 *Let $\mathcal{H}$ be a planar subgraph of a graph $\mathcal{G}$ and let $\tilde{\mathcal{H}}$ be a planar embedding of $\mathcal{H}$. Then $\tilde{\mathcal{H}}$ is $\mathcal{G}$–admissible if $\mathcal{G}$ is planar and there is a planar embedding $\tilde{\mathcal{G}}$ of $\mathcal{G}$ such that $\tilde{\mathcal{H}} \subseteq \tilde{\mathcal{G}}$.*

Definition 2.1.11 *A bridge in a graph is an edge whose removal would increase the number of components.*



(a) $\mathcal{G}$ (b) $\tilde{\mathcal{G}}$ (c) $\mathcal{H}$ (d) $\tilde{\mathcal{H}}$

Figure 2.2: An example of planar embedding and admissibility of graph $\mathcal{G}$

In figure 2.1 the graphs 2.1(a) and 2.1(c) are planar and the graph 2.1(b) is not. An example of planar embedding and admissability of a graph presents figure 2.2, where the graph $\tilde{\mathcal{G}}$ is a planar embedding of $\mathcal{G}$, the graph $\mathcal{H}$ is $\mathcal{G}$–admissible and the graph $\tilde{\mathcal{H}}$ is $\mathcal{G}$–inadmissible. If B is any bridge of $\mathcal{H}$ (in $\mathcal{G}$), then B is said to be *drawable* in a face f of $\tilde{\mathcal{H}}$ if the vertices of attachment of B to $\mathcal{H}$ are contained in the boundary of f. We write $F(B, \tilde{\mathcal{H}})$ for the set of faces of $\tilde{\mathcal{H}}$ in which B is drawable. The following theorem provides a necessary condition for $\mathcal{G}$ to be planar [BM76].

Theorem 2.1.12 *If $\tilde{\mathcal{H}}$ is $\mathcal{G}$–admissible then, for every bridge B of $\mathcal{H}$,*

$$F(B, \tilde{\mathcal{H}}) \neq \varnothing.$$

Since a graph is planar if and only if each block of its underlying simple graph is planar, it suffices to consider simple blocks. Given such a graph $\mathcal{G}$, the algorithm determines an increasing sequence $\mathcal{G}_1$, $\mathcal{G}_2$, ... of planar subgraphs of $\mathcal{G}$, and corresponding planar embeddings $\tilde{\mathcal{G}}_1$, $\tilde{\mathcal{G}}_2$, ... When $\mathcal{G}$ is planar, each $\tilde{\mathcal{G}}_i$ is $\mathcal{G}$–admissible and the sequence $\tilde{\mathcal{G}}_1$, $\tilde{\mathcal{G}}_2$, ... terminates in a planar embedding of $\mathcal{G}$. At each stage, the necessary condition in theorem 2.1.12 is used to test the planarity of $\mathcal{G}$ [BM76].

Algorithm 2.1.13 *Planarity Algorithm [BM76]*

1. *Let $\mathcal{G}_1$ be a cycle in $\mathcal{G}$. Find a planar embedding $\tilde{\mathcal{G}}_1$ of $\mathcal{G}_1$. Set $i = 1$.*
2. *If $E(\mathcal{G}) \backslash E(\mathcal{G}_i) = \varnothing$, stop (because all edges have been considered and $\mathcal{G}$ is planar). Otherwise, determine all bridges of $\mathcal{G}_i$ in $\mathcal{G}$; for each such bridge B find the set $F(B, \tilde{\mathcal{G}}_i)$.*

3. *If there exists a bridge B such that $F(B, \tilde{\mathcal{G}}_i) = \varnothing$, stop (because $\mathcal{G}$ is nonplanar). If there exists a bridge B such that $|F(B, \tilde{\mathcal{G}}_i)| = 1$, let $\{f\} = F(B, \tilde{\mathcal{G}}_i)$. Otherwise, let B be any bridge and f any face such that $f \in F(B, \tilde{\mathcal{G}}_i)$.*

4. *Choose a path $P_i \subseteq B$ connecting two vertices of attachment of B to $\mathcal{G}_i$. Set $\mathcal{G}_{i+1} = \mathcal{G}_i \cup P_i$ and obtain a planar embedding $\tilde{\mathcal{G}_{i+1}}$ of $\mathcal{G}_{i+1}$ by drawing P_i in the face f of $\tilde{\mathcal{G}}_i$. Replace i by $i+1$ and go to step 2.*

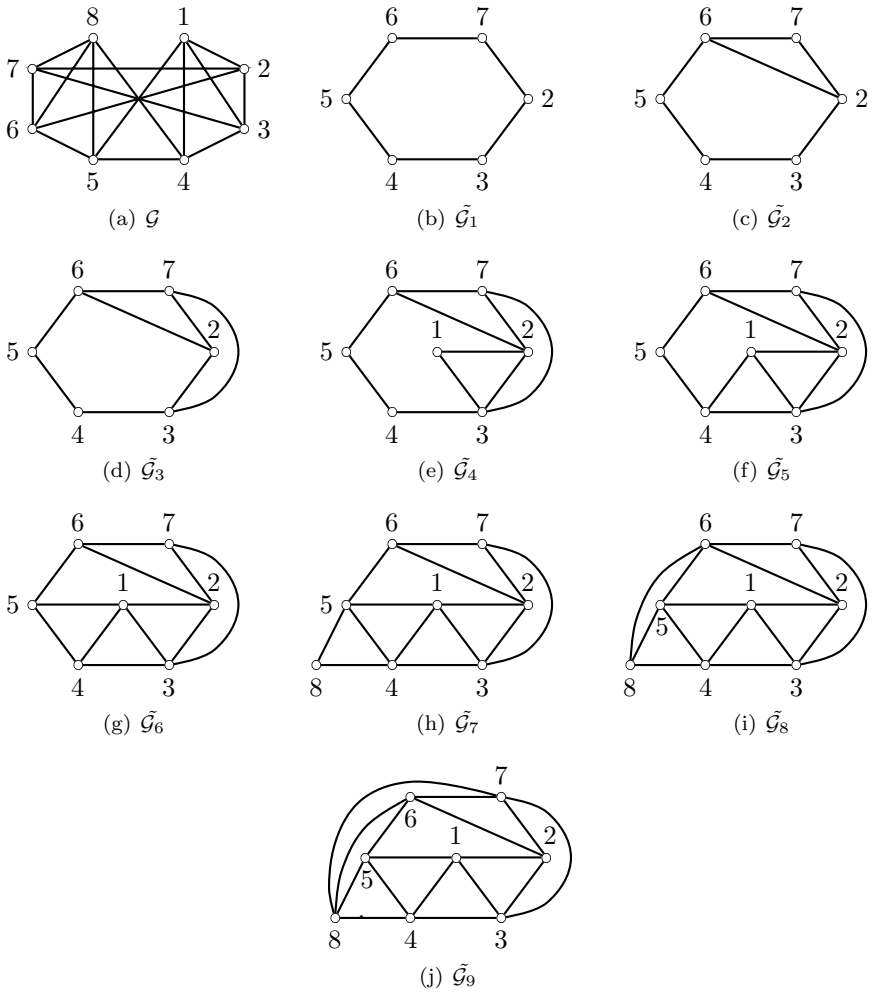


Figure 2.3: Example of using the planarity algorithm

An example of using the algorithm 2.1.13 for the graph $\mathcal{G}$ is illustrated in figure 2.3 [BM76]. It starts with the cycle $\tilde{\mathcal{G}}_1 = 2345672$ and a list of its bridges

(denoted, for brevity, by their edge sets). At each stage, the bridges B for which $|f(B, \tilde{\mathcal{G}}_i)| = 1$ are indicated in bold face:

(a) The graph $\mathcal{G}$;

(b) $\tilde{\mathcal{G}}_1$ and {12,13,14,15},{26},{48,58,68,78},{37};

(c) $\tilde{\mathcal{G}}_2$ and {12,13,14,15},{**48,58,68,78**},{**37**};

(d) $\tilde{\mathcal{G}}_3$ and {**12,13,14,15**},{**48,58,68,78**};

(e) $\tilde{\mathcal{G}}_4$ and {**14**},{**15**},{**48,58,68,78**};

(f) $\tilde{\mathcal{G}}_5$ and {**15**},{**48,58,68,78**};

(g) $\tilde{\mathcal{G}}_6$ and {**48,58,68,78**};

(h) $\tilde{\mathcal{G}}_7$ and {**68**},{**78**};

(i) $\tilde{\mathcal{G}}_8$ and {**78**};

(j) $\tilde{\mathcal{G}}_9$.

In this example, the algorithm terminates with a planar embedding $\tilde{\mathcal{G}}_9$ of $\mathcal{G}$, *i.e.*, $\mathcal{G}$ is planar.

Definition 2.1.14 *Two graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ are isomorphic if there is a one-to-one correspondence between the vertices of $\mathcal{G}_1$ and the vertices of $\mathcal{G}_2$ such that the number of edges joining any two vertices in $\mathcal{G}_1$ is equal to the number of edges joining the corresponding two vertices in $\mathcal{G}_2$*

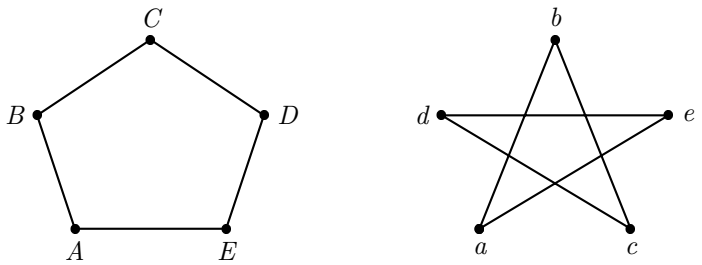


Figure 2.4: Two isomorphic graphs

The two graphs in figure 2.4 are isomorphic, because there is a correspondence between their vertices:

$$A \leftrightarrow a, B \leftrightarrow b, C \leftrightarrow c, D \leftrightarrow d, E \leftrightarrow e.$$

The task of checking on isomorphisms requires a lot of computer resources (time, memory) because it is necessary to test all permutations of vertices and their connections (for two graphs of N points it is $N!$). Therefore, one of the heavy problems is to avoid testing as much as possible. This can be done by using graph invariants to partition the vertices into equivalence classes. For example, no two vertices of different degree can be mapped to each other in an isomorphism, so the set of degrees defines equivalence classes. Observe that since any simple graph except $\mathcal{K}_1$ ($\mathcal{K}_n$ is the complete graph on n vertices) contains at least two vertices of equal degree, the degree sequence is not sufficient to completely partition the vertices of any graph [KST94].
Below necessary conditions for isomorphism of two graphs are presented [Tru94].
Properties preserved by isomorphism:

1. *The number of vertices.*
   Two isomorphic graphs have the same number of vertices;

2. *The number of edges.*
   Two isomorphic graphs have the same number of edges;

3. *The distribution of degrees.*
   In isomorphic graphs corresponding vertices have the same degree, because isomorphism preserves vertex adjacency;

4. *The number of "pieces" of a graph.*
   Isomorphism preserves the number of pieces (disconnected parts) of a graph, that is, isomorphic graphs are composed of the same number of pieces.

Two graphs sharing all four properties are frequently isomorphic, and whether they are isomorphic or not, can be determined by a careful search. Below we give an example of two graphs which preserve all four properties but are not isomorphic [Tru94].

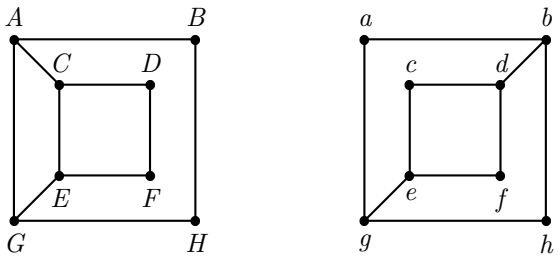


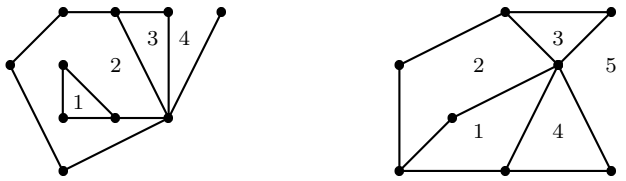
Figure 2.5: Two non–isomorphic graphs

Example 2.1.15 *The graphs of figure 2.5 are non–isomorphic, even though they share the four properties: in each $V = 8$ and $E = 10$; each has four vertices of degree two and four of degree three; and they are both in one piece. The structure difference is that the vertices of degree two are not related in the same way in the two graphs. In the first graph the vertices of degree two come in adjacent pairs: B is adjacent to H and D is adjacent to F. But in the second graph they are completely separated from one another by vertices of degree three: no vertex of degree two is adjacent to any other vertex of degree two. To see how this prevents the graphs from being isomorphic, we shall systematically try to construct an isomorphism in all possible ways. We begin by noticing that B, being of degree two, would have to correspond to a, c, f, or h.*

- *First case: B corresponds to a.*
  Then H, being adjacent to B, would have to correspond to b or g. But H must correspond to a vertex having the same degree, so H cannot correspond to either b or g. Therefore no isomorphism is possible if B made to correspond with a.
- *Second case: B corresponds to c.*
  Then H, being adjacent to B, would have to correspond to either d or e. But H must correspond to a vertex having the same degree, so H cannot correspond to either d or e. Therefore no isomorphism is possible if B is made to correspond to c.
- *Third case: B corresponds to f.*
  Then H, being adjacent to B, would have to correspond to either d or e, which is impossible as we noted in the second case. Therefore no isomorphism is possible if B is made to correspond to f.
- *Fourth case: B corresponds to h.*
  Then H, being adjacent to B, would have to correspond to either b or g, which is impossible as we noted in the first case. Therefore no isomorphism is possible if B is made to correspond to h.

We have seen that no isomorphism is possible if B is made to correspond to a, c, f or h. But under any isomorphism, B would have to correspond to a, c, f or h. Therefore no isomorphism is possible, and the graphs of figure 2.5 are not isomorphic [Tru94]. □

From definition 2.1.8 we know that a planar graph is a graph which can be drawn without edge–crossings. The edges of such a graph cut the plane into a number of regions which are called *faces*. The faces can be numbered and the number of all faces is denoted by F. An example of faces for planar graphs is given in figure 2.6. The graph of figure 2.6(a) has $F = 4$ and the graph of figure 2.6(b) has $F = 5$ (we must consider the external face of the graph too).



(a) (b)

Figure 2.6: Faces of graphs

Definition 2.1.16 *A graph is polygonal if it is planar, connected, and has the property that every edge borders on two different faces.*

Clearly, the graph in figure 2.6(a) is not polygonal whereas the graph of figure 2.6(b) is polygonal. For all polygonal and planar graphs the well-known *Euler's formula* holds (the proof can be found in [Tru94]):

Theorem 2.1.17 *If a graph $\mathcal{G}$ is planar then*

$$V - E + F = 2 \tag{2.1.1}$$

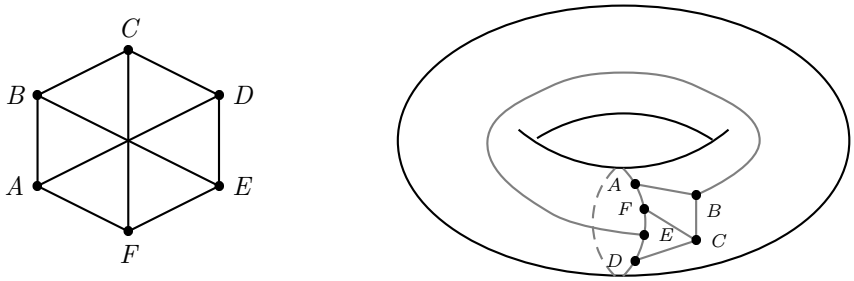
Euler's formula helps to prove many theorems in Graph Theory for planar graphs. One of them that we will use in this thesis is:

Theorem 2.1.18 *If a graph $\mathcal{G}$ is planar then $\mathcal{G}$ has a vertex of degree ≤ 5.*

Proof.
Case1. Let $\mathcal{G}$ have less than three vertices. Then it must be $\mathcal{K}_1$ or $\mathcal{K}_2$ and the theorem holds.
Case2. Suppose $\mathcal{G}$ has N vertices and the degree of all vertices is ≥ 6. Then we can number the vertices of $\mathcal{G}$ from 1 to N and write the series of statements:

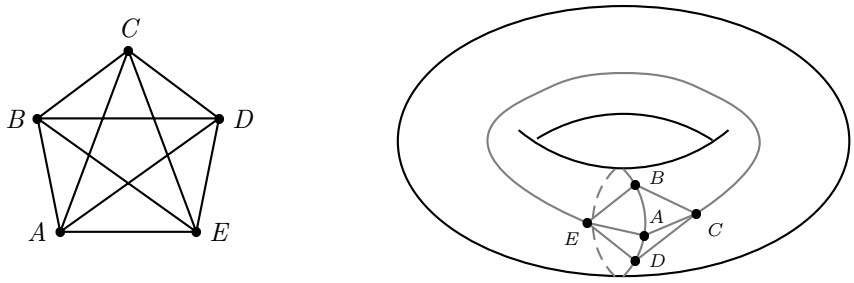
$$\{6 \ \leq \ d(v_i)\}_{i=1}^{N}$$

By theorem 2.1.2 one knows that $\sum_{i=1}^{N} d(v_i) = 2E$ and by supposition $d(v_i) \geq 6$. Combination of both statement gives $6V \leq 2E$ and after simplifying and applying Euler's formula (2.1.1) it becomes $3V \leq 3V - 6$, what is impossible. Therefore our assumption is incorrect. □
An extension of Euler's formula holds for a non–planar graph. We know that a graph is planar if, firstly, it can be drawn without edge–crossings and, secondly, at the same time it can be drawn in a plane. If we have another kind of surface, for example an one–hole torus (see figures 2.7(b) and 2.8(b)), it is possible to draw a graph on it without edge–crossing, but after projection in a plane it



(a) (b)

Figure 2.7: The non–planar graph is drawn on torus without edge–crossings



(a) (b)

Figure 2.8: The non–planar graph is drawn on torus without edge–crossings

will be non–planar. Examples of graphs which are planar on the one–hole torus but not on the plane are given in figures 2.7(a) and 2.8(a). The concept which includes the first condition for planar graphs but generalizes the second one by considering graphs drawn on other surfaces is called *genus* [Tru94].

Definition 2.1.19 *The genus of a graph, denoted "g", is the subscript of the first surface among the family S_0, S_1,..., on which the graph can be drawn without edge–crossings. (S_n is a n–hole torus)*

Euler's formula for a connected graph with genus g is

$$V - E + F = 2 - 2g \tag{2.1.2}$$

As one can note the formula differs from Euler's formula for a planar graph only by $2g$: a planar graph can be drawn on a **2D** sphere without edge–crossings and the genus for the sphere is zero.
In Graph Theory there are also other types of graphs, which are called *regular* and *platonic* graphs [Tru94]:

Definition 2.1.20 *A graph is regular if all the vertices have the same degree. If the common value of the degrees of a regular graph is d, we say that the graph is regular of degree d.*

Definition 2.1.21 *A graph is platonic if it is polygonal and regular, and has the additional property that all faces have the same number of edges.*

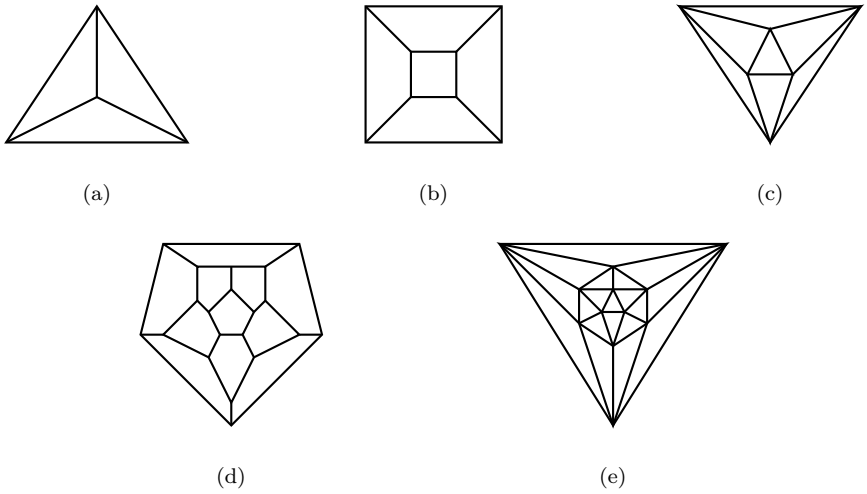


Figure 2.9: Platonic graphs

There are only five platonic graph: tetrahedron, cube, octahedron, dodecahedron, icosahedron (see figure 2.9(a)– 2.9(e) resp.). If we present them as three–dimensional drawings of graphs they become the regular polyhedron [Tru94].

Definition 2.1.22 *A regular polyhedron is a polyhedron - a closed solid figure bounded by a finite number of planes - having congruent regular polygons for its faces and all of its corner angles the same size.*

Certainly there are many other polyhedra which are not regular. In present thesis we only consider polyhedra with triangular faces, which are called triangulations. In the next section we give definitions and basic notions of them. We also show that a triangulation on the sphere is a planar graph with triangular faces.

2.2 About triangulation

Triangular meshes or triangulations provide an essential step in the spatial analysis of the data and are commonly used for representing surface in **3D**. Given

a set of vertices sampled from a source (in general, smooth) surface, triangular mesh with these vertices serves as a representation (approximation) of the sample surface. The quality of the approximation obviously depends on the particular choice of the triangulation. On the base of the triangulation a piecewise linear interpolating surface is uniquely determined [Alb03].
The following definition of triangulation is standard:

Definition 2.2.1 *A triangulation T is a collection of triangles, that satisfies the following properties:*

1. *Two triangles are either disjoint, or have one vertex in common, or have two vertices and consequently the entire edge joining them in common;*
2. *T is connected.*

We are interested only in compact surface[2]. In this case a triangulation T consists of a finite number of triangles, and the following two conditions are valid [Mas91]:

1. Each edge is an edge of exactly two triangles;
2. For every vertex v of a triangulation T, we may arrange the set of all triangles with v as a vertex in cycle order $t_0, t_1, \ldots, t_{n-1}, t_n = t_0$, such that t_i and t_{i+1} have an edge in common for $0 \leq i \leq n-1$.

The last condition means, that our triangulated surface is a manifold, *i.e.*, the neighborhood of every point, as well as a vertex, is topologically the same as the open unit ball in $\mathbb{R}^3$. The first condition of definition 2.2.1 excludes any intersection. Actually, the above–mentioned definition is the definition of an abstract triangulation of an abstract surface. Real surfaces can have self–intersections and/or have singularities, in which a surface is not a manifold. Therefore we distinguish between the topological abstract representation of (triangulated) surface S (as a 2-dimensional manifold) and its realization in 3–space (as an image of this "canonical" manifold) [Alb03].

Definition 2.2.2
1. *The union of all the faces and edges that contain the vertex is called the star of v;*
2. *For every vertex v of a triangle of T, the edges opposite v in the triangles $t_0, t_1, \ldots, t_{n-1}$ of T are called the links of the star.*

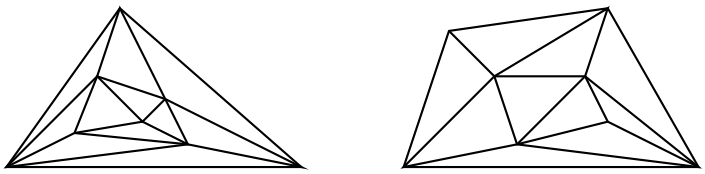
Given the data set, we can construct many triangulations depending on the connections between vertices. We assume that each triangulation preserves main topological characteristics, such as genius and orientability, of the underlying

[2]The main definitions of a surface you can find in section 7.1.

source surface. In the present thesis we deal only with "simple polyhedra", *i.e.,* triangulations which are topologically equivalent to a **2D** sphere. This equivalence is defined as follows:

Definition 2.2.3 *A continuous function $f : X \longrightarrow Y$ between two Euclidean sets is called a topological equivalent (homeomorphism) if it is one-to-one and onto, and if the inverse function $f^{-1} : Y \longrightarrow X$ is continuous.*

Informally speaking this means that such an object can be transferred by a continuous deformation to the **2D** sphere. In this work we reconstruct the surface with a triangulation which forms a closed polyhedron with triangular faces. For such a triangulation Euler's formula (2.1.1) holds, where V is the number of points (or vertices), E is the number of edges and F is the number of faces (triangular in our case). For a triangulation of N vertices the number of edges and triangles equals $E = 3(N-2)$ and $F = 2(N-2)$, respectively.
In topological graph theory under a triangulation on a sphere one understand a simple graph G embedded on the surface so that each face is triangular and any two faces share at most one edge [Neg91]. Thus, a triangulation can be presented by a planar graph which has one triangle as the outer face. For example, let us consider a triangulation on the sphere. We can cut out any triangular face and draw the rest in the plane in such a way that the remaining vertices and edges are inside this triangular face. The resulting object (see figure 2.10(a)) is one big triangle (outer face) which corresponds to the cut-out triangular face, and contains the remaining faces of the triangulation without internal intersections. It means that such a planar graph is a representation of a triangulation on a **2D**



(a) (b)

Figure 2.10: A planar graph is a triangulation (a) and is not a triangulation (b)

sphere. Actually the graph is polygonal with all internal and external faces as triangles. But not every polygonal graph is a representation of a triangulation on a **2D** sphere. For example, we take a few points in the plane and construct some polygonal graph on them. Let all internal faces be triangles but the external not (see figure 2.10(b)). It means that such a triangulation cannot be a representation of a triangulation on the sphere. Indeed, we can redraw this graph in such a way that a triangulation will have the outer triangular face but

inside there is one non–triangular face (see figure 2.11). Therefore we consider only the subclass of polygonal graphs all faces of which, including the outer face, are triangles.

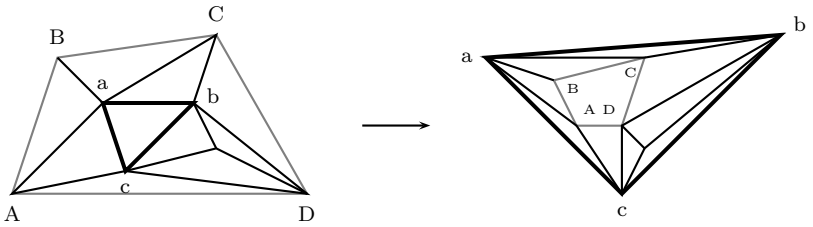


Figure 2.11: The polygonal graph which is not a triangulation on the sphere

Now there is enough information about graphs to proceed with the analysis and development of methods for generating triangulations. In the next chapter we estimate an upper bound and exponential approximation of the number of all possible triangulations and give two recursive methods for generating all possible non–isomorphic triangulations.

Remark 2.2.4 *As can be noted, three platonic polyhedra, namely tetrahedron, octahedron and icosahedron, have triangular faces and will be considered as initial triangulations for constructing all possible non-isomorphic triangulations in section 4.*

Remark 2.2.5 *If we have a graph $\mathcal{G}$ and we want to determine whether it is a representation of a triangulation of a closed object which is topologically equivalent to the sphere, or not, we need an adequate algorithm. The algorithm to check it, is:*

Algorithm 2.2.6

1. *Check planarity of the graph (see algorithm 2.1.13). If $\mathcal{G}$ is planar, goto step* 2*, otherwise stop the algorithm;*
2. *If $\mathcal{G}$ is polygonal, goto step* 3*, otherwise stop the algorithm;*
3. *Count the number of vertices V and edges E (or faces F). If $V = N$ and $E = 3(N-2)$ (or $F = 2(N-2)$), then $\mathcal{G}$ is a representation of a triangulation of the above type, otherwise it is not.*

PROOF. As was shown before, for a triangulation on the **2D** sphere Euler's formula holds, the number of edges and faces equal $3(N-2)$ and $2(N-2)$, respectively, (faces are triangles) and a triangulation can be represented as a planar graph. Following the algorithm 2.2.6, one has to check whether a graph of N vertices is planar, polygonal and $E = 3(N-2)$ (or $F = 2(N-2)$). Realization of these conditions means that the graph is a representation of a triangulation, because from (2.1.1) follows that $F = 2(N-2)$ (or $E = 3(N-2)$).□

Chapter 3

Generating "all" triangulations "once"

In the previous chapter we presented general theorems and definitions of graph theory, gave a definition of a triangulation and showed that a triangulation on the sphere is a polygonal (planar and connected) graph. In this chapter we use the properties of such a graph for generating a triangulation. We consider points without fixed coordinates. So we concentrate on the problem of generating a triangulation on the sphere as a topological problem; we do not care about possible self–intersections of the resulting surface as a whole.

The first study on the generation of all non-isomorphic triangulations was done by R. Bowen and S. Fisk [BF67]. They introduced three operations for adding a vertex of degree three, four and five to a triangulation in order to generate recursively a new triangulation, and they used checking on isomorphisms for extracting all non-isomorphic triangulations. Other studies such as [BP74, BT64, Mul65] mainly deal with enumerating triangulations. The problem of generating non-isomorphic triangulations was also considered by D. Avis in [Avi96] and by G. Brinkmann and B. McKay in [BM01]. D. Avis used a reverse search algorithm [AF92] and an edge swapping operation, whereas G. Brinkmann and B. McKay considered a triangulation as a certain type of graph that is imbedded on the surface of the sphere and they developed an algorithm for generating all of them.

The goal of this chapter is to consider triangulations in their totality, in order to develop different algorithms for generating them all, that can help us to study their properties. Therefore, we start from the outset. Firstly, we give an upper bound and exponential approximation for the number of all possible triangulations of N vertices. Next, we present a recursive method for generating the triangulations of $N + 1$ vertices from a triangulation of N vertices (the

method is similar to the method of [BF67], but it has some improvements). Afterwards, using knowledge of graph theory, we generate all non–isomorphic combinatorial triangulations t_N^* of N vertices and find their number. Then we present another method for generating all non-isomorphic triangulations which we call the "method of Cross-Sections".

3.1 Upper bound and e^α–approximation

In this section we evaluate a rough bound on the maximum number of all possible triangulations of N vertices. To do this we use the well-know fact of combinatorial theory that all possible choices of l points from k ($l \leq k$) is $\binom{k}{l} = \frac{k!}{l!(k-l)!}$. In our case any edge connects only two vertices of a triangulation; therefore the number of all possible pairs of two vertices is

$$\#pair(v_i, v_j) = \binom{N}{2},$$

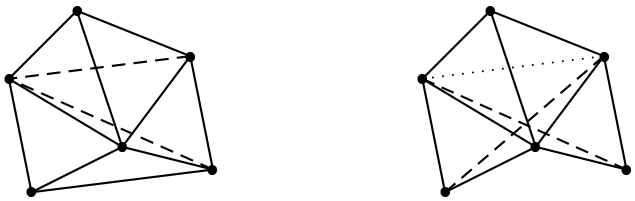
where $(i, j = 4, .., N)$ and $(i \neq j)$. But a triangulation of N vertices has only $3(N-2)$ edges and it is easy to see that

$$3(N-2) \leq \binom{N}{2},$$

where the equality holds only for $N = 4$. Therefore, an upper bound of the number of all possible triangulations t_N is

$$t_N \leq \binom{\binom{N}{2}}{3(N-2)}.$$

However, not all of these combinations are possible, because most of them do not satisfy the conditions in definition 2.2.1 starting from $N = 6$[1]. For example,



(a) (b)

Figure 3.1: (a) "Good" triangulation, (b) Forbidden "triangulation" (tessellation)

we take six points ($N = 6$) and try to make connections between all of them

[1]For $N = 4$ and $N = 5$ it is always possible to draw a planar graph with $3(N-2)$ edges so that all faces are triangles.

with $3(N-2) = 9$ edges (see figure 3.1). The graph in figure 3.1(a) indeed is a triangulation in the sense of definition 2.2.1, whereas the graph in figure 3.1(b) is *not* a triangulation because it has one edge (dotted), which is a common edge for three triangles.

Table 3.1 presents the numerical values of the upper bound and exponential approximation of the number of all possible triangulations t_N. It can easily be seen that the number grows very fast, approximately as $N^{3(N-4)2}$ and the exponential approximation is $A\alpha^N N^{(\beta N+\gamma)3}$. The treatment of such a large number of triangulations is a hard problem. This is why we focus in section 3.2 on the question of generating all triangulations t_N^* which are non–isomorphic from each other. It is clear that $t_N^* < t_N$. But indeed it also holds that $t_N^* \ll t_N$ as we will see later on.

N	$\binom{\binom{N}{2}}{3(N-2)}$	$N^{3(N-4)}$	$A\alpha^N N^{(\beta N+\gamma)}$	t_N^*
4	1	1	.101104E-1	1
5	10	125	.400974E+0	1
6	455	46656	.379350E+2	2
7	54264	.403536E+08	.712283E+4	5
8	.131231E+08	.687194E+11	.235090E+7	14
9	.556790E+10	.205891E+15	.125187E+10	50
10	.377365E+13	.100000E+19	.100929E+13	233
50	.139818E+192	.286985E+235	.110510E+192	—
100	.516574E+483	.100000E+577	.459930E+483	—
150	.163733E+810	.134267E+954	.151583E+810	—
200	.155606E+1159	.101306E+1354	.146880E+1159	—

Table 3.1: The upper bound and exponential approximation of t_N

[2]The function $f(N) = \binom{\binom{N}{2}}{3(N-2)}$ grows approximately as $N^{\alpha(N-4)}$ for large N. Therefore it is convenient to define $\alpha(N) = \frac{\ln(f(N))}{(N-4)\ln(N)}$ with $\lim_{N\to\infty} \alpha(N) = 3$.

[3]We use Maple to factorize $f(N) = \binom{\binom{N}{2}}{3(N-2)}$ by Stirling's formula $n! \sim \sqrt{2\pi} n^{(n+\frac{1}{2})} e^{-n}$ and find exponential approximation of $f(N) \sim A\alpha^N N^{(\beta N+\gamma)}$, where coefficients are $A = \frac{7776\sqrt{6}}{e^{12}\sqrt{\pi}}$, $\alpha = \frac{e^3}{226}$, $\beta = 3$, $\gamma = -\frac{13}{2}$.

3.2 Recursive method

In the previous section we found an upper bound and the exponential approximation of the number of all possible triangulations of N vertices. That number grows exponentially fast with N. In this section we present a recursive method for generating triangulations of $N+1$ vertices from a triangulation of N vertices. Furthermore, using the algorithm and properties of the planar graph, we generate all possible non–isomorphic triangulations and find how many they are.

3.2.1 Adding a new vertex to a triangulation (*Method A*)

In this subsection we want to develop a method for constructing a triangulation of N vertices, denoted with $T(N)$. The very simple way is to use recursion [oST]:

Definition 3.2.1 *An algorithmic technique where a function, in order to accomplish a task, calls itself with some part of the task is called recursion.*

Thus, by the definition 3.2.1 a triangulation of $(i+1)$ vertices can be constructed from a triangulation of i vertices by using the same operation(s) on every step of recursion $(i = 4, .., N-1)$.
Suppose we construct a triangulation of N vertices. Based on it we want to construct a new triangulation for $(N+1)$ vertices by adding an extra vertex. The Euler's formula for a triangulation of N vertices is:

$$V - E + F = 2. \tag{3.2.1}$$

and, therefore, for a triangulation of $(N+1)$ vertices is:

$$\widetilde{V} - \widetilde{E} + \widetilde{F} = 2, \tag{3.2.1'}$$

where $V = N$ and $\widetilde{V} = N+1$. The equation (3.2.1') can be simplified:

$$(N+1) - 3((N+1)-2) + 2((N+1)-2) = 2,$$
$$(N+1) - 3(N-1) + 2(N-1) = 2.$$

From the last equation it can be seen that after adding the $(N+1)$–st vertex to the triangulation of N vertices the number of edges grows by *three* and the number of faces grows by *two*. That is possible only if the new vertex is added to three vertices of the triangulation that belong to the same face, *i.e.*, $E_1 = E+3$ and in this case the number of faces increases by three (new faces) and decreases by one (internal face with three vertices), *i.e.*, $F_1 = F+3-1 = F+2$ (see figure 3.2(a)). Otherwise, a new triangulation of $(N+1)$ vertices will not be a

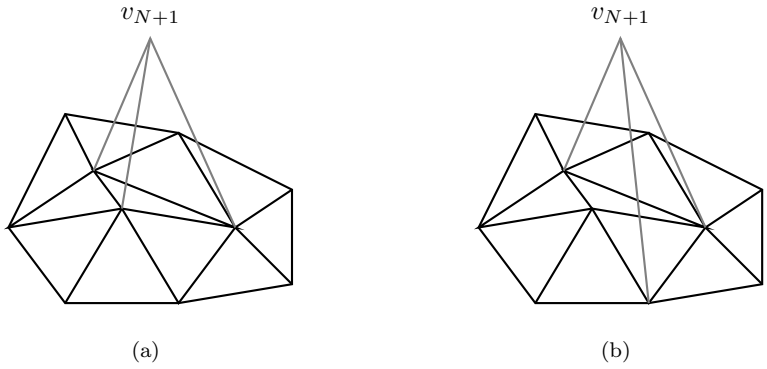


Figure 3.2: (a) Correct way to add a new vertex; (b) Wrong way to add a new vertex

triangulation in accordance with definition 2.2.1 (see figure 3.2(b)). Every face f to which we add a new vertex, has three adjacent faces f_1, f_2 and f_3. Deleting the common edge of f and f_i $(i = 1, 2, 3)$ forms a quadrilateral. By inserting a new $(N + 1)$–th vertex to this quadrilateral we obtain a new triangulation for $(N+1)$ vertices (see figure 3.3); and Euler's formula holds, because the number of vertices increases by one, $V_2 = N+1$; the number of edges increases by four and one internal edge (common edge of two faces) is deleted, $E_2 = E+4-1 = E+3$; and the number of faces increases by four and two internal faces are deleted, $F_2 = F + 4 - 2 = F + 2$. As a result $V_2 - E_2 + F_2 = 2$ becomes $N - E + F = 2$ after simplifying, which was required to be proved. As a next step we consider

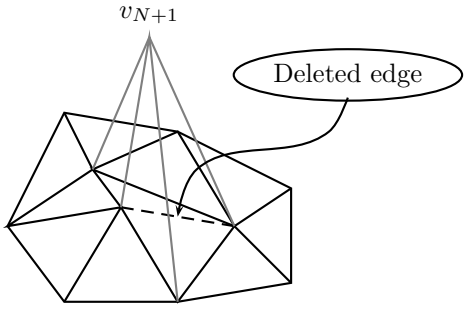


Figure 3.3: A new triangulation is constructed by deleting one edge

a quadrilateral and the adjacent faces to it. Deleting a common edge between a quadrilateral and one of the adjacent faces gives a pentagon which can be used for adding a new vertex for constructing a triangulation of $(N+1)$ vertices, and so on. The general scheme for the construction is:

Algorithm 3.2.2 *Method* A_{max}

Step 1.
Take a face f of a triangulation and add a new vertex to the three vertices of this face (the face f has three adjacent faces called f_1, f_2 and f_3);

Step 2.
Construct a quadrilateral P_4^i by deleting a common edge of f and f_i. Add a new vertex to the four vertices of P_4^i (quadrilateral P_4^i has four adjacent faces f_j, $i = 1..3$, $j = 1..4$);

Step 3.
. . . ;

Step K.
Construct a k-gon P_k^i by deleting a common edge of P_{k-1} and f_j. Add a new vertex to the k vertices of P_k^i (P_k^i has k adjacent faces f_j, $i = 1..k-1$, $j = 1..k$);

Step K+1.
. . . .

Lemma 3.2.3 *Euler's formula holds at every step of algorithm 3.2.2.*

PROOF. Suppose at step K there is a polygon P_k with k vertices, $(k-3)$ internal edges and $(k-2)$ internal faces. After adding the $(N+1)$–st vertex to it the new triangulation will have $V_{k-2} = N+1$, $E_{k-2} = E+k-(k-3) = E+3$ and $F_{k-2} = F+k-(k-2) = F+2$. Then Euler's formula for such a triangulation will be $V_{k-2}-E_{k-2}+F_{k-2} = 2$, which after simplifying, becomes $N-E+F = 2$. That was required to be proved.□
Note that the polygon must be a closed simple polygon, *i.e.,* it has no points inside. Otherwise after adding a new vertex to a polygon with, for example, one vertex v inside (as shown in figure 3.4), a new triangulation will have the edge vv_{N+1} inside itself, which contradicts the definition of a triangulation.

Theorem 3.2.4 *The recursive method A_{max} constructs all possible triangulations of $(N+1)$ vertices from triangulations of N vertices.*

PROOF. Suppose some triangulation $T(N+1)$ cannot be constructed by the *algorithm 3.2.2.* Let us take a vertex v of degree k which belongs to $T(N+1)$. If we cut off the vertex v with its links of the star V (see definition 2.2.2) then the adjacent to v vertices will form a closed simple polygon P_k on k vertices and the new combinatorial object will have N vertices, $3((N+1)-2)-k$ edges and $2((N+1)-2)-k$ faces. Because we deal with triangulations we should triangulate the polygon P_k (see remark 3.2.5 below). After triangulating polygon P_k, $(k-3)$ new edges and $(k-2)$ new faces are constructed. As a

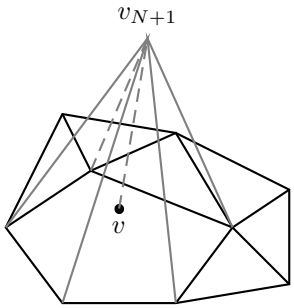


Figure 3.4: A new point is added to a polygon with one vertex inside

result, the numbers of edges and faces then become $3(N-2)$ and $2(N-2)$ respectively, *i.e.*, we get a triangulation $\widetilde{T(N)}$. Hence, by cutting off any vertex of degree k of $T(N+1)$ and triangulating the formed polygon P_k, we construct a triangulation of N vertices. But the operation "cutting off a vertex" is reverse to the operation "adding a new vertex" to a closed simple polygon with k vertices of triangulation $\widetilde{T(N)}$, therefore $\widetilde{T(N)}$ is one of the triangulations of N vertices. Consequently, the triangulation $\widetilde{T(N)} = T(N)$ was one of the triangulations used while constructing triangulations $T(N+1)$ by the algorithm. This is in contradiction with the assumption. □

Remark 3.2.5 *In the proof of theorem 3.2.4 we use a triangulating polygon P_k. The number of all possible triangulations of a polygon with $(n+2)$ sides is well–known and is called the Catalan number [MAT]:*

$$C_n = \frac{1}{n+1}\binom{2n}{n},$$

so that $C_0 = 1$, $C_1 = 1$, $C_2 = 2$, $C_3 = 5$, $C_4 = 14$ etc. For the polygon P_k the Catalan number is

$$C_{k-2} = \frac{1}{k-1}\binom{2(k-2)}{k-2}.$$

Therefore, cutting off any vertex of degree k of T^{N+1} gives C_{k-2} triangulations T^N. On the other hand using them for adding a new vertex of degree k gives C_{k-2} identical triangulations. Therefore the number of identical triangulations t_{N+1} that are constructed from triangulations of N vertices is $\sum_{k=3}^{K^*} C_{k-2}$, where $K^* = max_{v \in T_N}(d(v))$.

The *method* A_{max} constructs a new triangulation $T(N+1)$ from a triangulation $T(N)$ by adding a new vertex to a polygon on k vertices which has no points

inside. In graph theory such polygon is called a closed simple polygon or an empty cycle of a graph. It is possible to calculate the number of new triangulations that are constructed from a triangulation $T(N)$ for the first three steps, when a new vertex is added to a triangle, a quadrilateral and a pentagon:

- **adding a vertex to a triangle:** $t_{N+1}(P_3) = 3(N-2)$;
- **adding a vertex to a quadrilateral:** $t_{N+1}(P_4) = 2(N-2)$;
- **adding a vertex to a pentagon:** $t_{N+1}(P_5) = \sum_{\{v|d(v)\neq 3\}} n_v d(v)$;

where $d(v)$ is degree of vertex v and n_v is the number of vertices of degree $d(v)$. For the further steps ($k \geq 6$) this task of fixing t_{N+1} is much more complicated (some information about the upper bound of the number of all possible empty cycles can be found in Remark 3.2.8). But it is not necessary to perform these steps. Theorem 2.1.18 says that for any planar graph (a triangulation of the **2D** sphere) there is at least one vertex of degree three, four or five. Therefore we can assume that the added vertex has precisely one of those degrees. So we define only three operations whose application to all possible triangulations with N vertices yields all triangulations with $(N+1)$ vertices. These operations consist of adding a new vertex of degree three, four or five only to an appropriate triangulation with N vertices provided that minimum degree of all vertices of a new triangulation $T(N+1)$ equals three, four or five respectively. Adding a new vertex is then done according to the rules shown in figure 3.5. In case 3.5(c) the degree of the vertex v_2 must be at least six because after adding a new vertex its degree decreases by one.

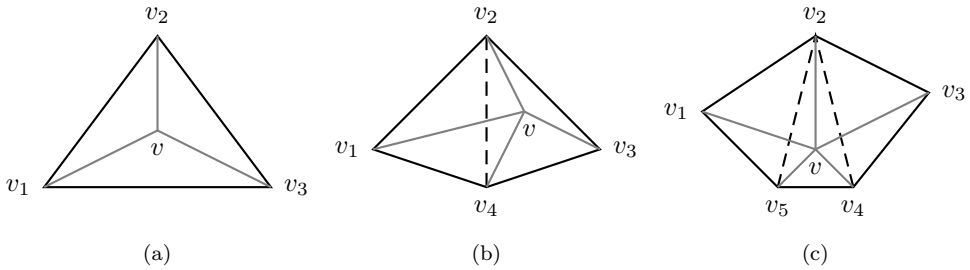


Figure 3.5: Adding a new vertex v of degree three (a), four (b) and five (c)

Below we present an algorithm for generating triangulations of $(N+1)$ vertices from a triangulation $T(N)$ by adding only vertices of degree three, four and five.

Algorithm 3.2.6 *Method A_3*

1. *For a triangulation $T(N)$, make a list of all faces:*
   $BL^3 = (P_1^3, .., P_{2(N-2)}^3)$, where P_i^j is an empty polygon of j vertices and i is the number of all possible such empty polygons for $T(N)$;

2. *Add a new vertex to each empty polygon of the list* BL^3*;*

3. **for** $k := 1$ **to** 2 **do***;*
   begin

   4. *Make a new list* BL^{k+3} *by deleting common edge(s) of* $P_i^{(k-1)+3}$ *and every adjacent face such that the new polygons* $P_{i'}^{k+3}$ *are be closed simple polygons;*

   5. *Add a new vertex to each empty polygon of the list* BL^{k+3}*;*

   end;

Theorem 3.2.7 *The recursive method* A_3 *constructs all possible triangulations of* $(N+1)$ *vertices from triangulations of* N *vertices.*

PROOF. The theorem holds because the *method* A_3 is a special case of the *method* A_{max}.□
In this section we have considered a recursive method for constructing triangulations of $(N+1)$ vertices from triangulations of N vertices. We represent every triangulation $T(N)$ as a graph and find all closed empty cycles on k vertices. Adding a new vertex to each of these cycles gives a new triangulation. The result of the method is presented in the Table 3.2, where $t_N^{A_{max}}$ and $t_N^{A_3}$

N	$t_N^{A_{max}}$	$t_N^{A_3}$	t_N^*
4	1	1	1
5	10	10	1
6	195	27	2
7	5712	134	5

Table 3.2: The result of methods A_{max} and A_3

are the numbers of all triangulations for N vertices[4], which are constructed by *method* A_{max} and *method* A_3 respectively, and t_N^* is the number of all non–isomorphic triangulations. Indeed, the number of triangulations which are constructed by these two methods is growing considerably faster than the number of non–isomorphic triangulations t_N^*. Therefore in the next subsection, we generate only non-isomorphic triangulations. It means that after constructing all

[4]In part II we consider triangulations on a real data set, where the number of triangulations are much less because a new vertex can be added only to closed simple polygons which are "visible" from it. "Visible" means that the edges and faces of a new vertex and vertices of a closed simple polygon do not intersect with the triangulation.

triangulations t_N^A (by *method* A_{max} or *method* A_3) we perform a check on isomorphisms (see definition 2.1.14 and example 2.1.15) between all of them and keep only non-isomorphic triangulations t_N^* for the next step of the algorithm, *i.e.,*

$$\cdots \rightarrow (t_N^*) \rightarrow t_{N+1}^A \rightarrow (t_{N+1}^A)^* = t_{N+1}^* \rightarrow \ldots .$$

As a result, the solution of Table 3.2 for *method* A_3 is changed by Table 3.3.

N	$(t_N^*)^{A_3}$	t_N^*
4	1	1
5	4	1
6	9	2
7	21	5

Table 3.3: The result after performing the check on isomorphism in method A_3

In this subsection we present recursive *method A* for generating triangulations of $(N+1)$ vertices from triangulations of N vertices. *Method* A_{max} and *method* A_3 are similar; the difference lies only in the number of steps, *i.e.,* the first method uses all possible empty cycles of a triangulation for adding a new vertex, whereas the second method uses only empty cycles on three, four or five vertices and has some restrictions on them for adding a new vertex. Because the number of triangulations to which a new vertex is added still grows very fast the concept of isomorphism was applied. As a result, in the remaining part of chapter 3 we will generate only non-isomorphic triangulations t_N^*.

Remark 3.2.8 *The problem of finding all closed simple polygons is a problem of graph theory which is called the problem of finding all empty cycles of a graph [BM76, Die99]. An upper bound of the number of all possible empty cycles of a planar graph of N vertices is given in 1997 by H. Alt, U. Fuchs and K. Kriegel [AFK99]:*

$$C(N) \leq 2^{2N-4} = \frac{4^N}{16}$$

In algorithm 3.2.2 we solve this problem numerically. As a result, we see that in practice the number of all possible empty cycles is about one half of value $C(N)$*:*

N	$C(N)$	Practical result
4	26	10
5	59	27
6	133	76
7	300	165

3.2.2 Method of indexing vertices (*Method B)*

In the previous subsection we considered a recursive method for generating triangulations. Because the number of all possible triangulations is very large we decided to use the concept of isomorphism, *i.e.*, consider only non–isomorphic triangulations for generating new ones. In chapter 2 we saw that the task of checking on isomorphism for two triangulations of N vertices requires testing of all permutations of vertices and their connections, that is $N!$. Thus, if we have k triangulations we need to make $\frac{k(k-1)}{2}$ tests on isomorphisms between every pair of them. As a result, to define all non–isomorphic triangulations we need $\frac{k(k-1)}{2}N!$ operations, thus the operation of checking on isomorphisms turns out to be computationally too expensive. Therefore, we want to avoid checking on isomorphisms as much as possible. For that, in this subsection, we introduce a technique of indexing vertices which uses the concept of automorphisms of graphs (triangulations in our case) and allows us to reduce the number of tests on isomorphisms between two triangulations.

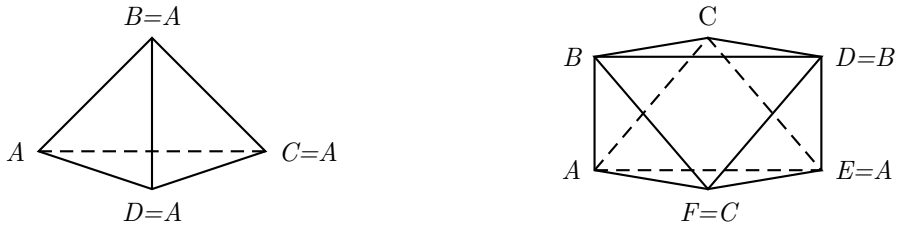


Figure 3.6: Automorphism of triangulations

Definition 3.2.9 *An automorphism of a graph is an isomorphism with itself.*

Let us look at the triangulations in figure 3.6. The triangulation on the left is completely automorphic. It means, that interchanging any two vertices gives

the same triangulation. Therefore, every vertex of a triangulation can be indexed with an identical letter. Vertices have identical label only if they are automorphic. As a result, the automorphism of faces of a triangulation can be defined too by using the labels of three vertices. Adding a new vertex to each automorphic face gives new isomorphic triangulations. It means that if we have two automorphic faces f_i and f_j of a triangulation $T(N)$ the adding a new vertex to f_i and f_j gives two isomorphic triangulations $T_i(N+1)$ and $T_j(N+1)$. Because of this there is a one-to-one correspondence between all faces in case of an automorphism for $T(N)$ and f_i automorphic to f_j. After adding a new vertex to f_i and f_j the rest of the faces of $T(N)$ will be the same for $T_i(N+1)$ and $T_j(N+1)$ and the one-to-one correspondence between them is preserved. Adding a new vertex to a face gives three new faces. A one-to-one correspondence can then only exist between three faces of $T_i(N+1)$ and $T_j(N+1)$.

Figure 3.6(a) shows a triangulation that has all vertices with identical label A. As a result, all faces have identical labels and they are automorphic to each other. Therefore adding a new vertex to any of the four faces gives four isomorphic triangulations. Thus, it is sufficient to add a new vertex to only one face. In figure 3.6(b) the triangulation has three pairs of vertices *A and E*, *B and D* and *C and F* with identical labels *A*, *B*, *C* respectively. Thus, the faces *ABC*, *EDC*, *ABF* and *EDF* have an identical label, just as the faces *AEC* and *AEF* and faces *BDC* and *BDF*. We define three types of non-automorphic faces *ABC*, *AEC* and *BDC*. Adding a new vertex to them gives three non-isomorphic triangulations, because adding a new vertex to faces of the same type, for example to *ABC* and *EDC* , gives isomorphic triangulations. Hence, it is sufficient to add a new vertex to three faces *ABC*, *AEC* and *BDC* instead of adding it to all eight faces.

By the method of indexing vertices, a triangulation $T_i(N)$ $(i = 1..t_N^*)$ gives only non-isomorphic triangulations $T_m^i(N+1)$ (because we add a new vertex only to non–automorphic faces of $T_i(N)$), where $m = 1..l_i$ and l_i is the number of non–isomorphic triangulations of $(N+1)$ vertices, which are generated from a triangulation $T_i(N)$. Thus, we need to make $l_i l_j$ tests on isomorphisms between two groups of triangulations $\{T_{m_1}^i(N+1)\}_{m_1=1}^{l_i}$ and $\{T_{m_2}^j(N+1)\}_{m_2=1}^{l_j}$ which are constructed from different triangulations $T_i(N)$ and $T_j(N)$ $(i, j = 1..t_N^*$, $i \neq j)$. To make a test on isomorphisms between all groups of new triangulations $\{T_m^i(N+1)\}_{m=1}^{l_i}$ $(i = 1..t_N^*)$ we need to consider $\sum_{i=1}^{I-1}\sum_{j=i}^{I} l_i l_j$ combinations $(I = t_N^*)$, whereas, as we discussed in the beginning of this subsection, in the *method A* we consider $\frac{k(k-1)}{2}$ combinations. Let us note, that in *method B* $k' \leq k$ and $k' = \sum_{j=1}^{I} l_j$ (k and k' are the numbers of all triangulations of $(N+1)$ vertices which are generated from triangulations of N vertices by the A and B respectively). In this way we reduce the number of tests on isomorphisms

and the result of the *method B* is presented in Table 3.4 (t_N^B were not computed for N equal to 11 and 12).

N	$t_N^{A_3}$	t_N^B	t_N^*
4	1	1	1
5	10	2	1
6	27	4	2
7	134	20	5
8	638	134	14
9	3396	1006	50
10	22028	9436	233
11	185313	—	1249
12	1756329	—	7595

Table 3.4: The number of triangulations of methods A_3 and B

3.2.3 Research by R. Bowen and S. Fisk

We obtained a recursive technique for generating triangulations. A similar method was developed by R. Bowen and S. Fisk in 1967 [BF67]. They developed an algorithm for constructing all (non–isomorphic) triangulations of the 2–sphere with N vertices from those with $(N-1)$ vertices. They considered Euler's formula (2.1.1) and the next theorem:

Theorem 3.2.10

$$\sum_{v \in N} V_{d(v)} d(v) = 2E,$$

where $d(v)$ is degree of vertex v and $V_{d(v)}$ is the number of vertices with this degree.

The formula (2.1.1) and the above equality yield the following expression:

$$\sum_{v \in N} V_{d(v)} (6 - d(v)) = 12, \tag{3.2.2}$$

which can be interpreted in the following way: for any triangulation of the 2–sphere there is at least one vertex with degree three, four or five, because the left side of the equation must be positive. On the basis of (3.2.2) Bowen and Fisk introduced three operations whose applications to all possible triangulations with $(N-1)$ vertices yield all triangulations with N vertices. So they add a new vertex of degree three, four and five in the same manner as it is described in subsection 3.2.1.

In our work we repeat their result but we also develop an improvement to it. The difference with their method consists of the following. We use special indexing vertices/faces to generate non-isomorphic triangulations of $(N+1)$ vertices from a triangulation of N vertices without checking on isomorphisms. This technique needs only a check on isomorphisms between groups of new non-isomorphic triangulations, and by that we reduce the number of tests on isomorphisms.

3.2.4 Conclusion of section 3.2

In section 3.2 we developed a recursive method for generating all possible non–isomorphic triangulations of $N+1$ vertices from a triangulation of N vertices. In the method we used the strategy of finding all possible empty closed simple cycles in a planar graph, to which a new vertex is added, and introduced the technique of "indexing vertices" which allows us to construct locally non-isomorphic triangulations of $(N+1)$ vertices from every triangulation of N vertices without checking on isomorphisms. This technique needs only a check on isomorphisms between groups of new non-isomorphic triangulations.

3.3 Method of Cross-Sections (*Method C)*

In this section we present a method which requires much less checking on isomorphisms and is called "Method of Cross-Sections". Unfortunately, it does not construct all triangulations.

An intuitive idea behind the method consists of the following simple consideration. Every vertex of a triangulation has a degree. The degree is defined as the number of vertices of the empty cycle to which the vertex was connected by the rule of the previous method. If we cut off the vertex by a cutting plane α such that the vertex and the other vertices lie on different sides of α, then the cutting plane will intersect the edges adjacent to the vertex v like in figure 3.7(a). We consider triangulations as abstract geometric complexes, *i.e.,* vertices can freely be moved as we want, with condition that straight edges are preserved (only the length of edges might change). Thus, the cutting plane can contain vertices and edges which belong to the $star(v)$ as in figure 3.7(b). Hence, a triangulation

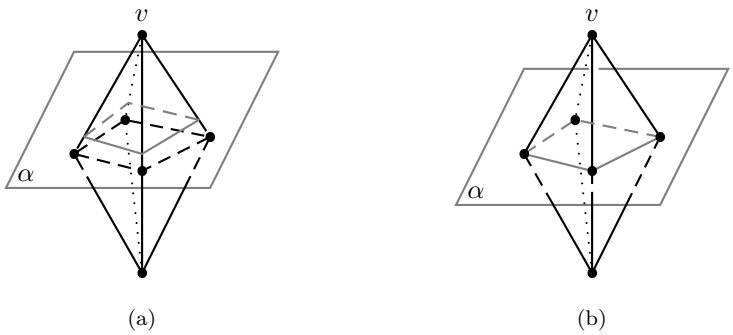


Figure 3.7: The cutting of a triangulation by plane α

can be split up in disjointed pieces by "cross-sections" that cut off one by one all vertices together with their stars. Finally each disjointed piece represents itself a polyhedral block which we call "simple" if it represents a k-pyramid or a "wedge"(see definitions below). This is an obvious fact but we would like to reduce the number of cross–sections.

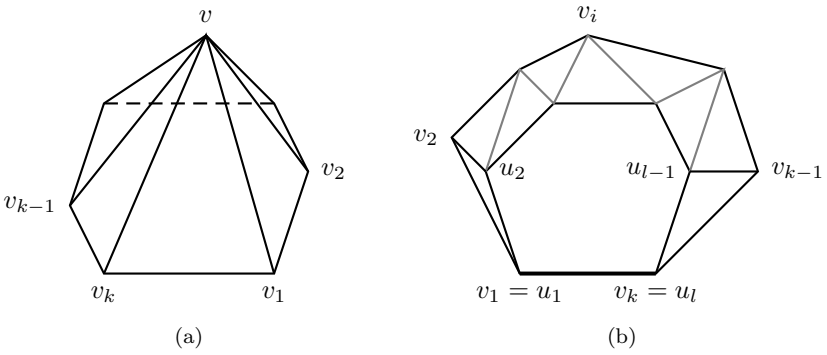


Figure 3.8: (a) k-pyramid, (b) wedge

Definition 3.3.1 *A "k-pyramid" is a polyhedron one face of which (known as the "base") is a k-gon $v_1...v_k$ and all the other faces are triangles meeting at a common polygon vertex v known as the "apex" (figure 3.8(a)).*

Definition 3.3.2 *A "wedge" is a polyhedron with two planar polygonal faces of k and l vertices with one common edge $v_1v_k = u_1u_l$, and with a triangulated polygon $v_2...v_{k-1}u_{l-1}...u_2$ (figure 3.8(b)).*

In figure 3.9 we present a triangulation which can be split up in simple polyhedral "blocks" (two pyramids an one wedge) just by two cross-sections: one

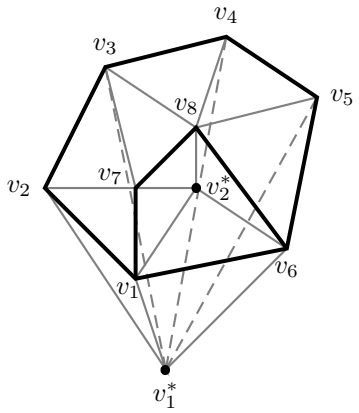


Figure 3.9: Two possible cross-sections of a triangulation

is $v_1v_2v_3v_4v_5v_6$ and the other is $v_1v_7v_8v_6$. Reversing the process of splitting we get a process of building a triangulation out of simple blocks. We state the following problem. Can we construct (build) any triangulation out of simple blocks?

Based on the above mentioned consideration we develop a method of cross-sections[5]. Let us introduce the terms we will use. A triangulation of N vertices, which can be cut with a plane such that two vertices of degree $N-2$ lie on different sides of it and the remaining $N-2$ vertices belong to the plane we call a triangulation with "Maximum Cross-Section". For example, the triangulation of figure 3.7(b) is a triangulation of six vertices with "Maximum Cross-Section" four. Faces, which are used for adding a new vertex, we call the "Main Cross-Section".

Our preposition is that we want to keep all our one–edge wedges, *i.e.*, any two cross-sections may have only one edge in common. Keeping this in mind we develop an algorithm.

Assume we have constructed a triangulation of four and five vertices and we want to continue the construction of triangulations for $N > 5$. There are two non–isomorphic triangulations of six vertices. One is constructed by adding a new vertex to the "Main Cross-Section" of the triangulation of five vertices (see figure 3.10(a)) and the other is constructed by adding the new vertex to the "Maximum Cross-Section" (see figure 3.10(b)). Repeating these operations to triangulations of six vertices and to the "Maximum Cross-Section" gives four triangulations of seven vertices. One triangulation is missing from this construction, however.

Let us consider the triangulation of six vertices. This triangulation was constructed by adding new vertices to the "Main Cross-Section" $v_2v_3v_4$ and then to

[5] An cross-section is a simple polygon.

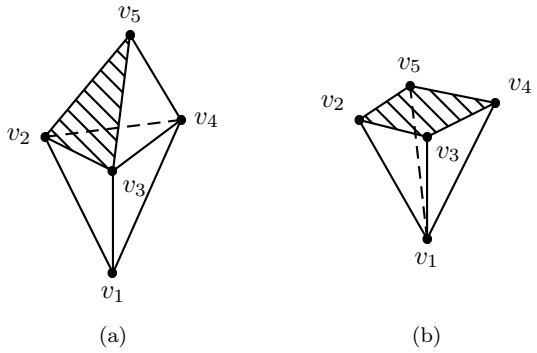


Figure 3.10: (a) "Main Cross-Section", (b) "Maximal Cross-Section"

the "Main Cross-Section" $v_2v_3v_5$ of the triangulations of four and five vertices respectively. If we draw in the plane the first "Main Cross-Section" $v_2v_3v_4$ and project v_5 and v_6 in this plane (figure 3.11(a)), then it is clear that the vertices v_5 and v_6 lie inside the triangle $v_2v_3v_4$. These vertices with two adjacent vertices of the triangle form a quadrilateral, for example $v_2v_3v_6v_5$ (figure 3.11(b)), which we call the "Potential Cross-Section". After adding a new vertex to it and triangulating the polygon $v_5v_6v_4$, the lost triangulation of seven vertices is found.

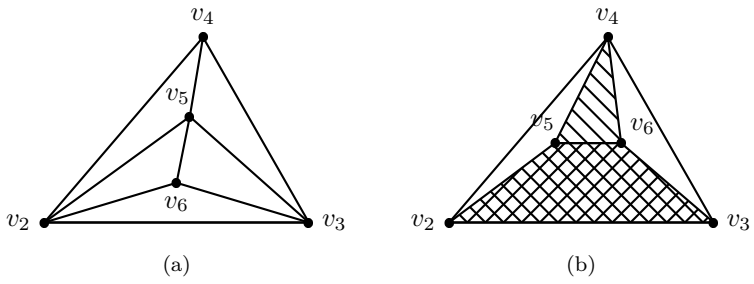


Figure 3.11: (a) projection of v_5 and v_6 on plane $v_2v_3v_4$ or "Main Cross-Section", (b) construction "Potential Cross-Section" with two vertices inside "Main Cross-Section"

Therefore, the construction of a triangulation consists of adding a new vertex to the "Main Cross-Section" (the same idea as in the previous section), to the "Maximum Cross-Section"(which is easy to construct) and to the "Potential Cross-Section". The "Potential Cross-Section" is constructed inside the "Main Cross-Section", the "Maximum Cross-Section" and the "Potential Cross-Section" of the previous triangulations by the next rules. For example, the triangulation of N vertices was constructed by adding m vertices one by one

to some "Maximum Cross-Section" of a triangulation of k vertices. We want to construct a triangulation of $(N+1)$ vertices. The next vertex is added to the "Potential Cross-Section" $v_1..v_m v_l v_{l+1}$ (see figure 3.12), where the vertices v_l and v_{l+1} belong to the "Maximum Cross-Section" and the rest vertices are inside the polygon ($m = N - k$). After adding the vertex it is necessary to triangulate the polygon $v_1..v_m v_l..v_1 v_k..v_{l+1}$. The vertices $v_1..v_m$ ($v_1..v_k$) must not be connected to each other and as a result, we have to triangulate the polygon $v_1..v_m v_{l-1}..v_1 v_k..v_{l+2}$, which has less vertices.

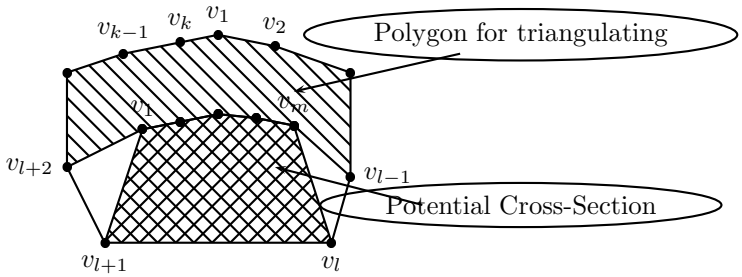


Figure 3.12: The construction of "Potential Cross-Section"

This method for the construction of a triangulation gives a result which can be found in Table 3.5, where t_N^C is the number of triangulations constructed by *Method C*, $t_N^{C^*}$ is the number of non–isomorphic triangulations after checking on isomorphism and t_N^* is the number of all non-isomorphic triangulations. This algorithm uses much less checking on isomorphisms because $t_N^C < t_N^B < t_N^{A_3}$. Unfortunately, this result does not coincide with the previous result, starting from nine vertices (observe in Table 3.5 that $t_9^{C^*} \neq t_9^*$). The problem lies in the

N	t_N^C	$t_N^{C^*}$	t_N^*
4	1	1	1
5	1	1	1
6	2	2	2
7	6	5	5
8	25	14	14
9	175	49	50

Table 3.5: The number of triangulation of the "Method of Cross-Sections"

fact that we always use only one edge of the "Cross-Section" for the construction of the "Potential Cross-Section". But in practice it is necessary to use two or more adjacent edges for the construction of all non-isomorphic triangulations (see remark 3.3.3 below), but then the method becomes much more complicated. An example of the triangulation of nine vertices which is constructed by using at least two edges of the "Potential Cross-Section" is shown in figure 3.13.

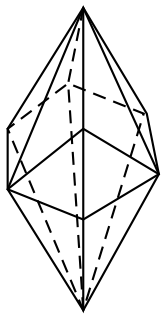


Figure 3.13: A triangulation which is not constructed by the "Method of Cross-Sections"

Remark 3.3.3 *Cycles are glued along one edge. We want to divide our triangulation by cycles such that the set of them can be divided in subsets so that the cycles, which belong to one subset, are glued along the same, unique, edge. Our proposition (any two cross-sections may have only one edge in common) turns out to be wrong, and two or more edge of wedges have to be glued along more than one edge. This can be explained by the following simple consideration. Consider a triangulation with many vertices and such that locally is a* 6*–regular graph. Taking an edge, we can define several cross-sections, but after that the next cross-section has to use an adjacent edge to the edge which we already used. Thus, a new cross-section will pass through two edges.*

Chapter 4

Generating non–isomorphic triangulations directly

In the previous chapter we considered recursive methods for generating all non-isomorphic triangulations. The basic idea behind the methods was to add one new vertex of degree three, four and five to a triangulation and then to select all non-isomorphic ones by applying testing on isomorphisms. The operation of checking on isomorphisms is computationally too expensive, therefore we want to avoid it as much as possible. Thus, we formulate the main problem of this chapter: "Is it possible to generate all combinatorial triangulations on the sphere without checking on isomorphisms?" The triangulations on the sphere are split into two subsets: one contains tree–like triangulations (*i.e.,* without cycles) and the second one contains cycles. Cycles create a problem, because we can get the same triangulation after generating two branches (which are coming together) of a non–isomorphic triangulation without checking on isomorphisms.

The first subset (tree-like triangulations) includes all triangulations which have at least one vertex of degree three. These triangulations are called dissectible and partly–dissectible. The second subset (triangulations with cycles) contains triangulations which have no vertices of degree three. They are also used as initial ones for constructing partly–dissectible triangulations.

Below, we present different methods for generating triangulations of the first and of the second subsets. One method generates all non–isomorphic triangulations of the first subset and does not need any checking on isomorphisms. The other method generates all non-isomorphic triangulations of the second subset. A certain class of the second subset is constructed without checking on isomorphisms. The remaining triangulations of the second subset are generated as well, but checking on isomorphisms is still needed.

4.1 Generating dissectible triangulations (GDT)

The problem of generating dissectible polyhedra (triangulations) is a natural generalization of the problem of dissecting a polygon, the history of which dates back to the 18-th century [Eul59]. W.G. Brown gave a historical survey and an extensive bibliography of dissections of a polygon in [Bro65]. Some approaches to the classical problem were done by J.W. Moon and L. Moser [MM63] and R.K. Guy [Guy58]. L.W. Beineke and R.E. Pippert in a series of papers [BP71, BP72, BP74] investigated enumeration of dissections of polygons and polyhedra by their automorphism groups. Other generalizations of the problem were done by W.G. Brown and W.T. Tutte [Bro63, BT64, Tut62, Tut63], by R.C. Mullin [Mul65] and by F. Takeo [Tak60, Tak61, Tak63, Tak64].
As we indicated above a dissectible polyhedron can be viewed as the **3D** analogue of the **2D** concept of triangulating a polygon using nonintersecting diagonals. The definition of a dissectible polyhedron has been given in the article "Enumerating dissectible polyhedra by their automorphism groups" of L.W. Beineke and R.E. Pippert [BP74]:

Definition 4.1.1 *1. A triangle and a tetrahedron are both dissectible polyhedra.*

2. A dissectible polyhedron with $(n+1)$ tetrahedra is obtainable from a dissectible polyhedron P with n tetrahedra by adding a new tetrahedron having precisely an exterior triangle in common with P.

In plain words this definition means that a triangulation is dissectible if it is possible to split it into separate tetrahedra, by removing every vertex by cutting off a corresponding tetrahedron, one by one. For example, all triangulations which are constructed by adding only vertices of degree three in the methods A and B of section 3.2 are dissectible. From another point of view, if one represents every tetrahedron of a dissectible triangulation as a node of a graph, then a dissectible polyhedron can be considered as a tree, *i.e.,* a graph without cycles. A dissectible polyhedron is also called a *tree–like triangulation* and it is a special case of 3–tree [BP74], where the definition of a 3–tree (2–tree) is, following [BP71]:

Definition 4.1.2 *A $(k-1)$–cell is a k–tree, and a k–tree with $n+1$ vertices is obtained when a k–cell is added to a k–tree with n vertices and has precisely a $(k-1)$–cell in common with it.*

The essential difference between a 3–tree and a dissectible polyhedron is that in a 3–tree a triangle can be shared by any number of tetrahedra, while in a dissectible polyhedron it can be shared by at most two [BP74]. Thus, the

problem of enumerating dissectible polyhedra is a special case of the above studies, and, others, in enumerating labelled trees, cubic graphs, dissecting k-balls.
In this section we consider the method, described in subsection 4.1.1, for generating all non-isomorphic dissectible polyhedra without the need of checking on isomorphisms. The main idea of the method is to add simultaneously l new vertices of degree three to a triangulation in a special way, such that it generates directly non-isomorphic triangulations. The method requires only knowledge on the automorphism of faces for the initial triangulation. L.W. Beineke and R.E. Pippert in their work [BP74] have used the idea of automorphism groups for enumerating dissectible polyhedra. They considered five possible types of permutations of the faces for a fixed tetrahedron:

i identity;

ii reflection;

iii diagonal rotation;

iv trigonal rotation;

v tetragonal rotation.

and three possible types of permutations of the faces for a non-fixed tetrahedron:

vi reversal;

vii half–turn;

viii hexagonal rotation.

Based on these types of permutations they defined seventeen types of automorphism groups for dissectible polyhedra. For each of them they calculated the exact number of polyhedra that admit such group, and found the general number of all unlabelled and unrooted dissectible triangulations. The other study about generating dissectible polyhedra was done by D. Avis in [Avi96]. He used the reverse search method, developed by himself and K. Fukuda [AF92], for generating all 2- and 3-connected r-rooted triangulations. The method which we present in this section differs from these two methods. In comparison with the work of L.W. Beineke and R.E. Pippert [BP74], where they enumerate dissectible polyhedra, we generate them all explicitly. In comparison with the work of D. Avis [Avi96], where he takes a triangulation and uses the operation of swapping edges for generating other triangulations, we use the operation of adding vertices in a special way, that guarantees the construction of all non-isomorphic dissectible triangulations without the need of checking on isomorphisms. This

way of constructing triangulations we will also use in part II of this work, where we consider generation of geometrical triangulations with fixed coordinates.
Let us note, that in this section we use as the initial triangulation the tetrahedron - one of the platonic graphs.

4.1.1 The main method of *GDT*

One of the most difficult problems in enumerating/generating combinatorial objects is the problem of controlling isomorphisms. For this reason many algorithms are restricted to generation of so-called rooted objects, where one "detail" of the object is taken to be fixed (for example, a vertex or an edge). The number of rooted objects is bigger than that of non–rooted ones, and some non-isomorphic rooted objects are still isomorphic in a usual setting. Most results have been obtained for rooted objects (maps, triangulations, etc.) because choosing a "root" destroys most of the symmetries, which makes enumeration/generation easier. However, if we want to derive from "rooted" objects all non–rooted ones, an additional check on isomorphisms is needed.
Indeed, if an object has no symmetries, we can presume that it does not matter what edge or vertex we take as a "root". However, if an object has symmetries, then some "roots" will be equivalent and therefore in the generation process isomorphic non–rooted objects will be produced. On the other hand, we cannot completely rule out the possibility of generating isomorphic objects at some step of the generating process even if the initial objects has no symmetries.
A heuristic idea is to use more "roots" at each step of the generating process, in order to avoid repetitions of the objects and the consequent testing on isomorphisms.
This idea leads to the logical conclusion: at each step of the generation process we should add not only one generating block at a time, but, depending on the situation, a collection of these generating blocks.
In our case the generating block is a vertex. We refer to vertices which we insert in some triangulation in order to produce new triangulations, as to *generating vertices*.
A triangulation (with N vertices), to which we add new generating vertices, is called a *preceding triangulation*, and triangulations, obtained from the preceding triangulation by inserting all possible collections of generating vertices, are called *descending triangulations*. The idea is to add simultaneously l new generating vertices to an initial/preceding triangulation with N vertices. As we show later the number l must satisfy the following condition:

$$v_3 \leqslant l \leqslant 2(N-2), \tag{4.1.1}$$

where v_3 is the number of vertices of degree three of the initial/preceding triangulation, and $2(N-2)$ is the number of all faces (triangles) of a triangulation

with N vertices. Of course, we cannot add more than $2(N-2)$ vertices to a triangulation with N vertices in one step. The lower bound $v_3 \leq l$ is an essential part of the algorithm. We explain it in details by describing the algorithm.
Indeed, the scheme of the algorithm is the following: new triangulations of $N+v_3$, $N+(v_3+1)$, ..., $N+(2N-4)$ vertices with v_3, v_3+1, ..., $2N-4$ vertices of degree three respectively are constructed from a triangulation of N vertices which has v_3 vertices of degree three (see the scheme in Table 4.1[1]).

		v_3				
N	Dt^*_N	2	3	4	...	$v_{3,max}$
5	$\{t(5,2)\}$	$\{t(5,2)\}$	—	—	...	—
6	$\{t(6,2)\}$	$\{t(6,2)\}$	—	—	...	—
7	$\sum_{i=2}^{3}\{t(7,i)\}$	$\{t(7,2)\}$	$\{t(7,3)\}$	—	...	—
8	$\sum_{i=2}^{4}\{t(8,i)\}$	$\{t(8,2)\}$	$\{t(8,3)\}$	$\{t(8,4)\}$	...	—
...	...	...	...	...	...	...
N	$\sum_{i=2}^{v_{3,max}}\{t(N,i)\}$	$\{t(N,2)\}$	$\{t(N,3)\}$	$\{t(N,4)\}$	...	$\{t(N,v_{3,max})\}$

Table 4.1: N is the number of vertices, Dt^*_N is the number of non-isomorphic dissectible triangulations of N vertices, v_3 is the number of vertices of degree three, $v_{3,max}$ is the maximum v_3, $\{t(N,v_3)\}$ is a group of all dissectible triangulations of N vertices with v_3 vertices of degree three

The main algorithm for generating directly all non–isomorphic triangulations of M vertices from all previous ones is:

Algorithm 4.1.3 *The main algorithm for GDT.*

1. *$T(4,1)$ is the tetrahedron, $Dt^*_4 = 1$;*
2. **for** $N := 4$ **to** $M-1$ **do**
   3. **for** $i := 1$ **to** Dt^*_N **do**
      begin
      4. *choose a triangulation $T(N,v_3)$ with the next representation:*
         a) *All faces $\{f_i\}$ $(i = 1,\ldots,2(N-2))$of $T(N,v_3)$ are given in the next order:*

[1] $v_{3,max}$ can easily be found for a triangulation of M vertices. Both equations $M = N+v_3$ and $v_{3,max} = 2(N-2)$ give $v_{3,max} = 2((M-v_3)-2) \geq 2((M-v_{3,max})-2)$. As a result, $v_{3,max} = \lceil \frac{2(M-2)}{3} \rceil$.

$\{f^1_{i+3(1-1)}\}^3_{i=1}$, ..., $\{f^{v_3}_{i+3(v_3-1)}\}^3_{i=1}$, $f_{3v_3+1},\ldots,f_{2N-4}$,
where $\{f^j_{i+3(j-1)}\}^3_{i=1}$ *is three faces of* j*–th star;*

b) Automorphism groups $Aut(T(N, v_3))$.

5. $l := v_3$*;*

6. **repeat**

7. **if** $l = v_3$

then *Find all possible combination of* v_3 *faces (each of them belongs to one star) in lexicographic order and write them into list* L_{v_3}*. The number of such combinations is* 3^{v_3}*;*

else $L_l := L^{'}_{l-1} + \{f_i\}$*, where* $\{f_i\} := F \backslash \{L^{'}_{l-1}\}$*, i.e.,* $\{f_i\}$ *are the remaining faces which have never been used in the list* $L^{'}_{l-1}$*;*

8. $L^{'}_l := reduct(L_l)$*;*

9. Add l *new vertices to* l *faces of a triangulation* $T(N, v_3)$ *from the list* $L^{'}_l$ *and generate new non-isomorphic triangulations* $T(N+l, l)$*;*

10. Find $Aut(T(N + l, l))$*;*

11. $l := l + 1$*;*

12. **until** $l > 2N - 4$ *or* $N + l > M$*;*

end;

We now clarify the subsequent steps of the algorithm.

- In *step 4* we have a triangulation $T(N, v_3)$ and we need to generate a triangulation $T(N + l, l)$ $(v_3 \leq l)$. For that we add l new vertices to the triangulation $T(N, v_3)$ in a special way. First, we consider all stars of degree three (each star has three faces) and find all possible combinations of v_3 faces in lexicographic order such that every face of a combination belongs to different stars. The maximum number of such combinations is 3^{v_3};

- In *step 7 and step 8* we find all possible combinations of l faces and write them in the list L_l. Then we apply the procedure $reduct(L_{v_3})$, which rejects all automorphic combinations from the list L_l and produces a new list $L^{'}_l$ of non–automorphic combinations of l faces (explanation is below), where the number of combinations of $L^{'}_l \leq$ the number of combinations of L_l;

- In *step 9, step 10 and step 11* we add l new vertices to all combinations of the list $L^{'}_l$ of $T(N, v_3)$. New triangulations $T(N + l, l)$ will be generated and for them we find the automorphisms $Aut(T(N + l, l))$. After that we increase the number of faces in $L^{'}_l$ by one, that gives a new list L_{l+1}, and repeat the same operations with L_{l+1}.

Remark 4.1.4 *Note that, for avoiding repetition of combinations of l faces in the list L_l, we always put them down in the lexicographic order.*

Step 8 of the algorithm uses procedure $reduct(L_l)$. Now we describe how it works. Suppose we have the group of automorphisms for an initial/preceding triangulation T of N vertices which are presented in Table 4.2 (group of automorphisms is denoted by $Aut(T)$ in the algorithm).

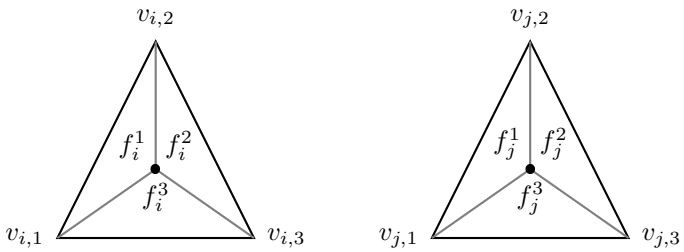
	f_1	f_2	f_3	$\ldots$	$f_{2(N-2)}$
1	$g_1(f_1)$	$g_1(f_2)$	$g_1(f_3)$	$\ldots$	$g_1(f_{2(N-2)})$
2	$g_2(f_1)$	$g_2(f_2)$	$g_2(f_3)$	$\ldots$	$g_2(f_{2(N-2)})$
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
p	$g_p(f_1)$	$g_p(f_2)$	$g_p(f_3)$	$\ldots$	$g_p(f_{2(N-2)})$

Table 4.2: Automorphism groups for a triangulation T

It is clear that adding a new vertex to faces f_i, $g_1(f_i)$, ..., $g_p(f_i)$ of T gives $p+1$ isomorphic triangulations, because all these faces are automorphic to each other (see subsection 3.2.2). Therefore it is necessary and sufficient to add a new vertex to only one of the faces. If we want to add simultaneously two new vertices to, for example, two faces f_i and f_j, then the new triangulation will be isomorphic to a triangulation, which is generated by adding simultaneously two new vertices to any of pair of faces $(g_1(f_i), g_1(f_j))$, $(g_2(f_i), g_2(f_j))$, ..., $(g_p(f_i), g_p(f_j))$. It means that it is necessary and sufficient to add only two new vertices simultaneously to, for example, faces f_i and f_j and *not* to any of the other combinations. The same idea, of course, is used for adding more new vertices simultaneously to T, such that new triangulations will be non–isomorphic. In accordance with the algorithm the list L_l contains combinations of l faces of $T(N, v_3)$. Using $Aut(T(N, v_3))$ we can define in L_l combinations of l faces which are automorphic to each other, therefore we choose only one of them and delete the others from the list L_l. Repeating the same operation produces a new (reduced) list $L_l^{'}$ which has only non–automorphic combinations of l faces.

Finally, we note that it seems obligatory to find a group of automorphisms for every descending triangulation. But this is not necessary, because a group of automorphisms can easily be found from the group of automorphisms of the preceding triangulation T. (Recall that every descending triangulation will be a preceding one in the next step (*step 10*). Suppose some combination $\{f_i\}_{i=1}^{l}$ be equal to the combination $\{g_s(f_i)\}_{i=1}^{l}$, where $1 \leq s \leq p$. Equality means

that all faces $\{f_i\}_{i=1}^{l}$ are contained in the sequence $\{g_s(f_i)\}_{i=1}^{l}$ but they can be in a different order. It means that such a combination is automorphic itself, *i.e.*, adding new vertices to T results in an automorphic triangulation. Determining a new group of automorphism is subjected to the next rules. Let a face f_i be automorphic to some face f_j of a triangulation T, where $f_j = g_s(f_i)$. Adding simultaneously new vertices to f_i and f_j produces three new faces for each of them. Let f_h^1, f_h^2, f_h^3 be the new faces which are constructed after adding a new vertex to face f_h $(h = i, j)$, see figure 4.1.



(a) face f_i (b) face f_j

Figure 4.1: Defining new group of automorphisms

Because f_i is automorphic to f_j, there will be a one–to–one correspondence between the new faces f_i^1, f_i^2, f_i^3 and f_j^1, f_j^2, f_j^3, such that, if edge $e(v_{i,h_1}, v_{i,h_1+1})$ corresponds to edge $e(v_{j,h_2}, v_{j,h_2+1})$ in the preceding triangulation T, then in the new (descending) triangulation face $f_i^{h_1}$ will correspond to $f_j^{h_2}$, where $h_1, h_2 = 1, \ldots, 3$ and $v_{*,4} = v_{*,1}$.

Remark 4.1.5 *If the new (descending) triangulation after adding l new vertices to the faces $\{f_i\}_{i=1}^{l}$ has no non–trivial group of automorphisms, then for such a triangulation in the next step new vertices can be added to all combinations of the full list L_l without reduction.*

In this subsection we presented an algorithm which generates all non–isomorphic dissectible triangulations without any checking on isomorphisms. The result of this generation is given in Table 4.3. The algorithm is not incremental in the first step when we add v_3 new vertices simultaneously. But in further steps it is incremental, because, after adding v_3 new vertices to a triangulation, the next vertices $((v_3 + 1)$–th, $(v_3 + 2)$–th, $\ldots)$ are added one by one.
In the next two subsections we explain the first steps of the method in more detail. Furthermore, we prove that all triangulations which are generated by algorithm 4.1.3 are non-isomorphic and that their generation is exhaustive.

		t_N^* with respective $v_3 =$							
N	Dt_N^*	2	3	4	5	6	7	8	t_N^*
5	1	1	—	—	—	—	—	—	1
6	1	1	—	—	—	—	—	—	2
7	3	2	1	—	—	—	—	—	5
8	7	4	2	1	—	—	—	—	14
9	24	10	11	3	—	—	—	—	50
10	93	25	48	19	1	—	—	—	233
11	434	70	209	138	16	1	—	—	1249
12	2110	196	857	852	195	10	—	—	7595
13	11002	574	3425	4891	1913	196	3	—	49566
14	58713	1681	13142	25640	15349	2786	114	1	339722
15	321776	5002	49268	126087	107301	31366	2702	50	2406841

Table 4.3: N is the number of vertices, Dt_N^* is the number of non-isomorphic dissectible triangulations of N vertices, v_3 is the number of vertices of degree three

4.1.2 The first steps of algorithm 4.1.3 in detail

1. **Generating new triangulations from the initial triangulation of four vertices:**

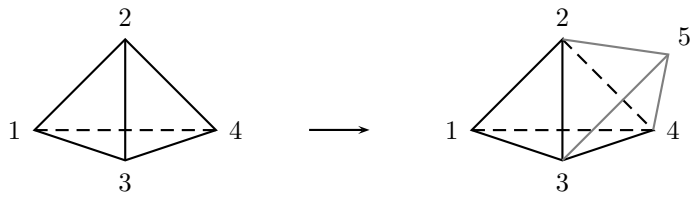


Figure 4.2: Adding one new vertex to the tetrahedron

As we mentioned above, there exists only one triangulation with four vertices: each of the vertices has degree three, and the triangulation has four faces. In $\mathbf{3}D$ representation this triangulation realizes as a tetrahedron and is the initial triangulation for algorithm 4.1.3. This is the only triangulation where stars of vertices with degrees three are "glued" together,

Id	f_1	f_2	f_3	f_4	9	f_2	f_3	f_1	f_4	17	f_3	f_4	f_1	f_2
2	f_1	f_2	f_4	f_3	10	f_2	f_3	f_4	f_1	18	f_3	f_4	f_2	f_1
3	f_1	f_3	f_2	f_4	11	f_2	f_4	f_1	f_3	19	f_4	f_1	f_2	f_3
4	f_1	f_3	f_4	f_2	12	f_2	f_4	f_3	f_1	20	f_4	f_1	f_3	f_2
5	f_1	f_4	f_2	f_3	13	f_3	f_1	f_2	f_4	21	f_4	f_2	f_1	f_3
6	f_1	f_4	f_3	f_2	14	f_3	f_1	f_4	f_2	22	f_4	f_2	f_3	f_1
7	f_2	f_1	f_3	f_4	15	f_3	f_2	f_1	f_4	23	f_4	f_3	f_1	f_2
8	f_2	f_1	f_4	f_3	16	f_3	f_2	f_4	f_1	24	f_4	f_3	f_2	f_1

Table 4.4: Group of automorphisms for the initial triangulation of four vertices

and in this case we can add maximally four vertices of degree three. All four vertices and all four faces of the initial triangulation are isomorphic to each other and the group of all automorphism of faces are presented in table 4.4, where $f_1 = 123$, $f_2 = 134$, $f_3 = 142$, $f_4 = 324$ (see figure 4.2). Therefore, it does not matter, to which face we add a new point. We choose one face and add a new vertex to it. The obtained triangulation will be a triangulation for five vertices with two vertices of degree three and this is also the only possible triangulation with five vertices (the group of automorphisms for that triangulation is given in table 4.6).

Id	f_1	f_2	f_3	$f_{4.1}$	$f_{4.2}$	$f_{4.3}$
2	f_1	f_3	f_2			
3	f_2	f_1	f_3			
4	f_2	f_3	f_1			
5	f_3	f_1	f_2			
6	f_3	f_2	f_1			

Table 4.5: Creating the group of automorphism for the triangulation of five vertices

For example, we add the fifth vertex to face f_4. Therefore we find all rows in table 4.4 such that $f_4 = g_p(f_4)$ and $p = 3, 7, 9, 13, 15$. Then we create

a new table of the group of automorphisms for the triangulation of five vertices and copy the rows $3, 7, 9, 13, 15$ for column f_1, f_2, f_3 as it is shown in table 4.5.

Id	f_1	f_2	f_3	f_4	f_5	f_6	7	f_4	f_5	f_6	f_1	f_2	f_3
2	f_1	f_3	f_2	f_4	f_6	f_5	8	f_4	f_6	f_5	f_1	f_3	f_2
3	f_2	f_1	f_3	f_5	f_4	f_6	9	f_5	f_4	f_6	f_2	f_1	f_3
4	f_2	f_3	f_1	f_5	f_6	f_4	10	f_5	f_6	f_4	f_2	f_3	f_1
5	f_3	f_1	f_2	f_6	f_4	f_5	11	f_6	f_4	f_5	f_3	f_1	f_2
6	f_3	f_2	f_1	f_6	f_5	f_4	12	f_6	f_5	f_4	f_3	f_2	f_1

Table 4.6: Group of automorphisms for the preceding triangulation of five vertices

Adding the fifth vertex to f_4 gives three new faces $f_{4.1} = 325$, $f_{4.2} = 435$ and $f_{4.3} = 245$ (see figure 4.2). In row 2 of table 4.5 we see that f_2 is automorphic to f_3 and f_1 is not. This means that $f_{4.2}$ is automorphic to $f_{4.3}$. Therefore we can write in row 2 of table 4.5 that $f_{4.2} \rightarrow f_{4.3}$, $f_{4.3} \rightarrow f_{4.2}$ and $f_{4.1} \rightarrow f_{4.1}$. The same operation is applied to the row $3, \ldots, 6$. Renaming $f_{4.1}$ by f_4, $f_{4.2}$ by f_5 and $f_{4.3}$ by f_6 gives the first six rows of table 4.6. The remaining six rows are constructed by interchanging the first triple column with the second triple column, because of symmetry of all four vertices of the initial triangulation. That operation is fulfilled only for this case.

We can also add two, three or four generating vertices to the initial triangulation (the tetrahedron) simultaneously. As a result, all descending triangulations with five, six, seven and eight vertices are generated[2]. These triangulations possess two, three or four vertices v_3. The process of generating descending triangulations from the initial preceding triangulation (the tetrahedron) is presented in figure 4.3. The group of automorphisms is found in the same manner as for the triangulation of five vertices.

2. **Generating new triangulations from the preceding triangulations of five vertices:**

   Next, we take the only possible triangulation of five vertices as preceding. This triangulation has six faces and two vertices of degree three. The group of all automorphism of faces are given in table 4.6 and are constructed

[2]These descending triangulations become then preceding in the next step.

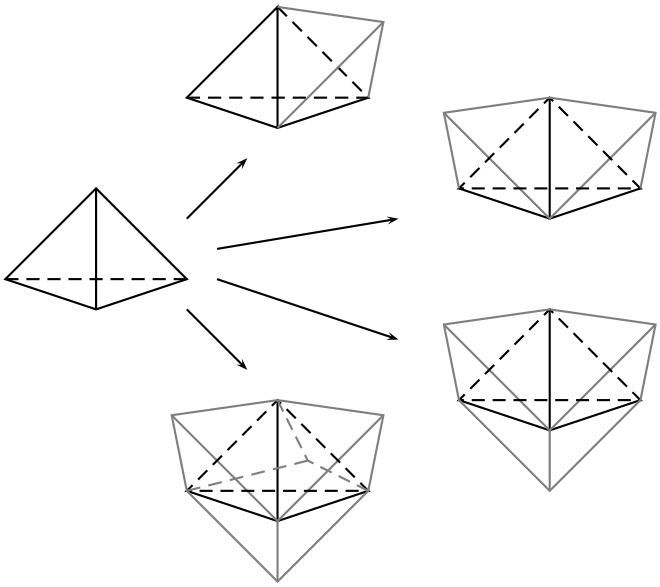


Figure 4.3: The initial triangulation of four vertices gives four new triangulations of five, six, seven and eight vertices

from the initial triangulation of four vertices, where $f_1 = 123$, $f_2 = 134$, $f_3 = 142$, $f_4 = 325$, $f_5 = 435$, $f_6 = 245$ (see figure 4.2).

According to our method we have to add at least two new vertices to this triangulation: each to one of the three faces of two active stars.

This can be performed in several ways (depending on the automorphism's of the preceding triangulation). Generating a triangulation of five vertices can be done in two ways, and we obtain two triangulations with seven vertices, two of which are of degree three. Then we can add the third, the fourth, the fifth and, finally, the sixth new vertex to other faces of the triangulation with five vertices.

After applying the above-mentioned procedure we construct triangulations with seven, eight, nine, ten and eleven vertices with two, three, four, five and six vertices of degree three respectively (see figure 4.4) and find their group of automorphisms.

4.1.3 Properties of algorithm 4.1.3

We can proof that algorithm 4.1.3 generate all dissectible triangulation precisely once:

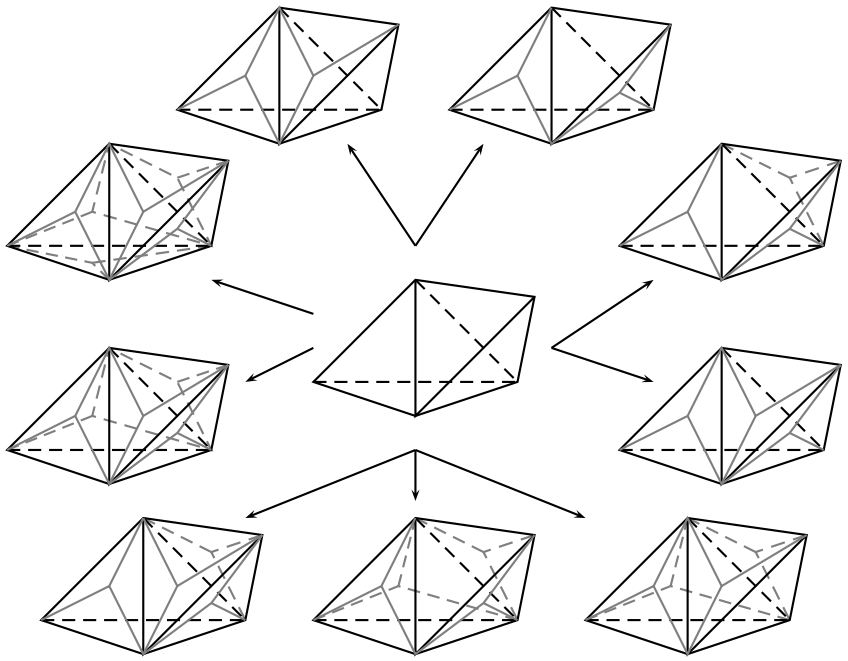


Figure 4.4: Adding two, three, four, five and six new vertices simultaneously to the preceding triangulation of five vertices

Theorem 4.1.6 *All generated triangulations with algorithm 4.1.3 are non-isomorphic and the generation is exhaustive.*

PROOF. All triangulations generated by performing the algorithm can be divided into different groups $\{t(N, v_3)\}$ in accordance with their number of vertices of degree three. The proof consists of four steps.

1. Each group has a different number of vertices of degree three and the checking on isomorphisms between any two triangulations from two different groups is not necessary by the definition of isomorphism. There cannot be a one-to-one correspondence between any triangulation with k_1 vertices of degree three and any triangulation with k_2 vertices of degree three, if $k_1 \neq k_2$.

2. Suppose now that two triangulations T_1' and T_2' belong to the same group but are generated from two non-isomorphic triangulations T_1 and T_2. Let us show that T_1' and T_2' are also non-isomorphic. Suppose each of these two triangulations has v_3 vertices of degree three $v_1^i, v_2^i, \ldots, v_k^i$ $(i = 1, 2)$ and they are isomorphic. It means that there is a one-to-one correspondence

between the vertex set v_i^1 of T_1' and the vertex set v_j^2 of T_2' $(i, j = 1, \ldots, k)$. If we delete vertices $v_1^1, v_2^1, \ldots, v_k^1$ from T_1' and vertices $v_1^2, v_2^2, \ldots, v_k^2$ from T_2', then the two obtained triangulations $\widetilde{T_1}$ and $\widetilde{T_2}$ must be isomorphic and $T_i = \widetilde{T_i}$ $(i = 1, 2)$ by construction, but by assumption T_1 and T_2 are not isomorphic. We obtained a contradiction.

3. Suppose triangulations T_1' and T_2' belong to the same group and have been generated from the same triangulation T by adding l vertices. By selecting all non–automorphism combinations of l faces for the triangulation T all new triangulations T_1' and T_2' will be non–isomorphic (procedure $reduct(L_l)$).

4. Finally, we show that we do not miss any triangulation. Suppose there is a triangulation $T(*, l_0)$ with l_0 vertices of degree three, but it does not belong to any group. By deleting the vertices of degree three we get a triangulation $T(*, l_1)$, where $l_1 \leq l_0$. If this triangulation belongs to some group $\{t(*, l_1)\}$ then triangulation $T(*, l_0)$ has to belong to some group $\{t(*, l_0\}$ by construction. Otherwise, we repeat the process of deletion of vertices of degree three. Suppose after m steps triangulation $T(*, l_m)$ obtained and $l_m \leqslant l_{m-1} \leqslant \ldots \leqslant l_1 \leqslant l_0$. Two cases are possible:

   (a) If $l_m \neq 0$ we continue the cutting–off till the triangulation of four or five vertices is obtained that contradicts our assumption.

   (b) If $l_m = 0$ then triangulation $T(*, l_m)$ has no vertices of degree three. But such a triangulation does not belong to the class of dissectible polyhedra.□

4.2 Generating partly–dissectible triangulations

In the previous section we considered the main algorithm 4.1.3 for generating all non–isomorphic dissectible triangulations without checking on isomorphisms. As it was said, we took the tetrahedron as the initial triangulation and constructed all dissectible non-isomorphic triangulations which belong to the first subset of triangulations.

In this section we consider how to generate the remaining triangulations of the first subset which are not dissectible but have vertices of degree three. These are all triangulations which we call "partly–dissectible". A triangulation is partly–dissectible if it is constructed by algorithm 4.1.3 **but has as an initial triangulation a triangulation of the second subset**, *i.e.,* a triangulation without any vertices of degree three. All dissectible triangulations are partly–dissectible but the contrary is false.

For generating partly–dissectible triangulations we use the same algorithm 4.1.3. But before starting the algorithm we have to find a triangulation of the second subset and its automorphism group. Thus, the problem of generating all possible non–isomorphic triangulations is reduced to generating all triangulations without vertices of degree three (triangulations of the second subset).
The result of generating all non–isomorphic partly–dissectible triangulations PDt^*_N without pure dissectible triangulations is presented in table 4.7.

		v_3					
N	PDt^*_N	1	2	3	4	t^*_N	Dt^*_N
7	1	1	—	—	—	5	3
8	5	2	3	—	—	14	7
9	21	7	11	3	—	50	24
10	128	33	62	27	6	233	93
11	781					1249	434

Table 4.7: N is the number of vertices, PDt^*_N is the number of non-isomorphic partly–dissectible triangulations of N vertices, v_3 is the number of vertices of degree three

4.3 Generating triangulations of the second subset

In the previous two sections we described the method for generating all possible non–isomorphic dissectible and partly-dissectible polyhedra without the necessity of any checking on isomorphisms. As we said above the generation of partly-dissectible triangulations needs a triangulation of the second subset as an initial one. Therefore in this section we present a method for generating all non–isomorphic triangulations of the second subset, which are represented themselves by 4–connected and 5–connected graphs (see definition 2.1.6).
R. Bowen and R. Fisk in their study [BF67] were the first who extracted the triangulations without vertices of degree three into a different class. They divided all triangulations into triangulations which are constructed by adding only vertices of degree three (dissectible and partly-dissectible) and the other ones (we call them triangulations of the second subset) and generated them up to 11 vertices. D. Avis and Ch.M. Kong in their work [AK96] generated rooted

triangulations with minimum degree four. They represented a triangulation as a rooted planar graph with minimum degree four. Then, by using a reverse search technique [AF92] they generated all non-isomorphic rooted triangulations with minimum degree four.

In this section we study triangulations of the second subset and present possible methods for constructing them. Finally, we adopt the idea of section 4.1, *i.e.*, adding a few new vertices per time to a triangulation, to generate triangulations of the second subset. For that we introduce the concept of "domain" for a triangulation. Loosely speaking, a domain is a part of a triangulation with a number of vertices of degree four. A triangulation consists of several disjoint domains. Adding simultaneously l new vertices of degree four to domains produces some class of non-isomorphic triangulations of the second subset without extra checking on isomorphisms.

In subsection 3.2.1 we showed that a triangulation without vertices of degree three can be constructed by adding a new vertex of degree four (or five) to a triangulation such that the new triangulation has no vertices of degree three (or four). In this section we only consider adding a new vertex of degree four. To do that we have to add a new vertex to two adjacent faces (triangles in our case) f_1 and f_2[3] and to delete their common edge $e(f_1, f_2)$ or equivalently, by inserting a new vertex into the adjacent edge $e(f_1, f_2)$ of two triangles f_1 and f_2 and connecting it to two vertices that are opposite to this edge (see figure 3.5(b)).

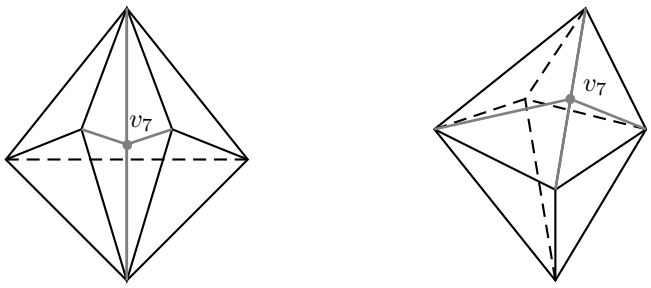
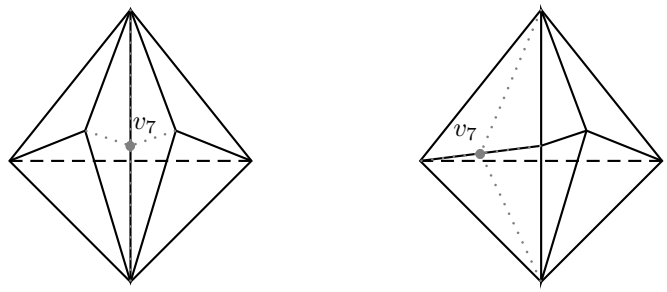


Figure 4.5: A triangulation of the second subset with seven vertices is constructed by adding a seventh vertex (gray dot) to a triangulation of six vertices which has: (a) two vertices of degree three; (b) no vertex of degree three

The first triangulation without vertices of degree three is the octahedron (one of the platonic graphs) which represents the triangulation of six vertices, that appears after adding a new vertex of degree four to the triangulation of five vertices

[3]We remind that adding a vertex v to a triangle t is equal to connecting v to three vertices of t.

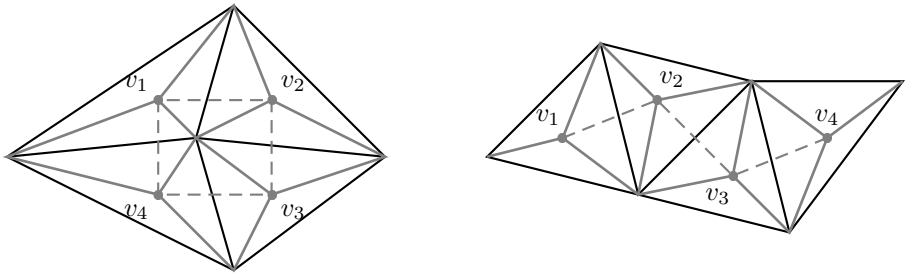
(dissectible triangulation with two vertices of degree three). For seven vertices we have also only one triangulation of the second subset which can be constructed from two different triangulations of six vertices. It can be constructed either by adding a new vertex of degree four to the dissectible triangulation with two vertices of degree three (see figure 4.5(a)) or by adding a new vertex of degree four to the triangulation of the second subsets, *i.e.*, without vertices of degree three (see figure 4.5(b)). The first construction is not needed because adding a vertex of degree four to triangulation with one or two vertices (necessarily adjacent) of degree three is equivalent to adding a vertex of degree to another triangulation without vertices of degree three (see figure 4.6).



(a) (b)

Figure 4.6: Adding a vertex of degree four to a dissectible triangulation with two vertices of degree three and to a triangulation of the second subset gives two equal triangulations

In section 3.2 we showed that the method of adding only one new vertex at each step for constructing all non-isomorphic triangulations of the second subsets needs checking on isomorphisms.



(a) (b)

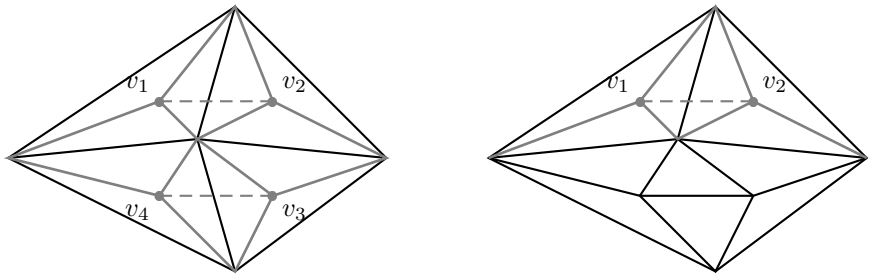
Figure 4.7: Constructing triangulations of the second subset by edge swapping

Another idea is to construct the triangulation with maximum cross-section for six, seven, eight,... vertices (see section 3.3) and then to add l vertices of degree three to each of them (l has to be more than one) and make all possible swaps of the common edge of two adjacent stars of degree three such that the new triangulation has no vertex of degree three. This can easily be done if the triangulation has only two or three adjacent stars of degree three. But it is difficult to define which edges have to be swapped if the number of adjacent stars is more than three. For example, in figure 4.7 the triangulations have four stars of degree three. There are seven possibilities to swap edges of the triangulation in the figure 4.7(a):

1. $(\{v_1, v_2\},\{v_3, v_4\})$
2. $(\{v_1, v_4\},\{v_2, v_3\})$
3. $(\{v_1, v_2\},\{v_3, v_4\}, \{v_2, v_3\}\ )$
4. $(\{v_1, v_2\},\{v_3, v_4\},\{v_1, v_4\})$
5. $(\{v_1, v_4\},\{v_2, v_3\},\{v_1, v_2\})$
6. $(\{v_1, v_4\},\{v_2, v_3\},\{v_3, v_4\})$
7. $(\{v_1, v_2\},\{v_2, v_3\},\{v_3, v_4\},\{v_4, v_1\})$;

and two possibilities for the triangulation in figure 4.7(b):

1. $(\{v_1, v_2\},\{v_3, v_4\})$
2. $(\{v_1, v_2\},\{v_3, v_4\},\{v_2, v_3\})$



(a) (b)

Figure 4.8: Swapping (a) two edges and (b) one edge gives two isomorphic triangulations

Unfortunately, the method for constructing triangulations of the second subsets by edge swapping in dissectible and partly–dissectible triangulations with at

least two vertices of degree three does not generate non–isomorphic triangulations (you can easily to see it in figure 4.8) therefore a checking on isomorphisms is still needed.

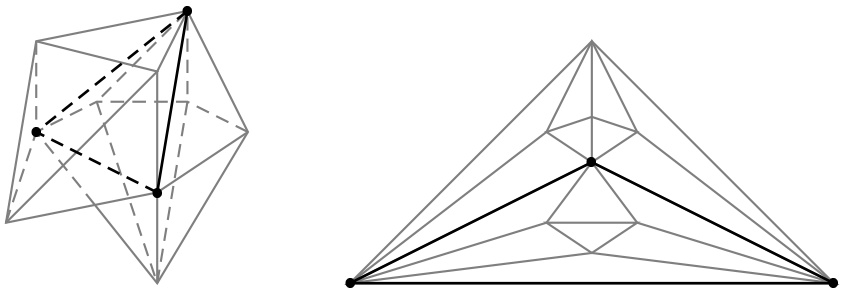


Figure 4.9: A triangulation is constructed by gluing through one face (black color) two triangulations of seven and six vertices

This method also does not generate triangulations, which can be constructed by gluing two triangulations through only one face, for example see figure 4.9. Therefore, we decide to divide the triangulations of the second subset into two types:

1. triangulations of the first type are triangulations which are constructed by gluing a number of triangulations through one face;
2. triangulations of the second type are the remaining triangulations of the second subsets which do not belong to the first type.

The construction of triangulations of the first type is not a difficult problem. For example, for constructing a triangulation $T(N)$ (a triangulation of N vertices) of the first type from two triangulations $T(K)$ and $T(L)$ $(3 < K, L < N)$ it is necessary to glue through one face triangulations $T(K)$ and $T(L)$ with the condition that $K + L - 3 = N$. For constructing $T(N)$ from l triangulations $T(K_i)$ $(i = 1..l)$ the next equality should hold $\sum_{i=1}^{l} K_i + 3(l-1) = N$. Of course, triangulations $T(K_i)$ and $T(K_j)$ should be glued through only one face $(i, j = 1..l)$. Note, that if $l > 2$ then dissectible and partly–dissectible triangulations can be used for generating new triangulations under only one condition that a new triangulation $T(N)$ must belong to the second subset, *i.e.,* has no vertices of degree three. An example of constructing a triangulation of the first type is presented in figure 4.10 and the number of them is given in table 4.8. Thus, constructing triangulations of the first type leads to solving the combinatorial problem of defining all different combinations of l triangulations and can be easily done[4]. The result is in table 4.8.

[4]We do not consider this problem in the present work.

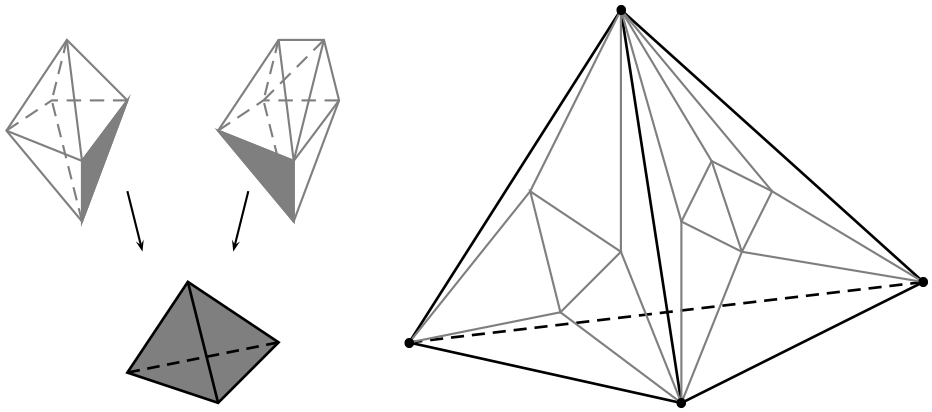


Figure 4.10: Tetrahedron $T(4)$ (black) is glued with $T(6)$ and $T(7)$ through one face to each of them that result in a triangulation of the first type $T(11)$

		St_N^*		
N	St_N^*	type one	type two	t_N^*
7	1	—	1	5
8	2	—	2	14
9	5	1	4	50
10	12	2	10	233
11	34	9	26	1249

Table 4.8: N is the number of vertices, St_N^* is the number of non-isomorphic triangulations of N vertices without vertices of degree three

Finally we consider the triangulations of the second type, the number of them is presented in table 4.8. The method for generating dissectible polyhedra uses stars of degree three and produces all non-isomorphic triangulations without checking on isomorphisms. By analogue we try to translate this method to a method for generating 4–connected triangulations, *i.e.,* instead of adding a vertex of degree three to a triangle we add it to a quadrilateral. However, the quadrilateral is triangulated in two ways, therefore the same triangulations will be produced. Thus, we have to distinguish such cases in order to avoid repetitions.

Let us start with the construction of triangulations from the octahedron. It

is the triangulation of six vertices which can be considered as a triangulation with six vertices of degree four. Inserting a new vertex into any edge gives a triangulation of seven vertices (see figure 4.11).

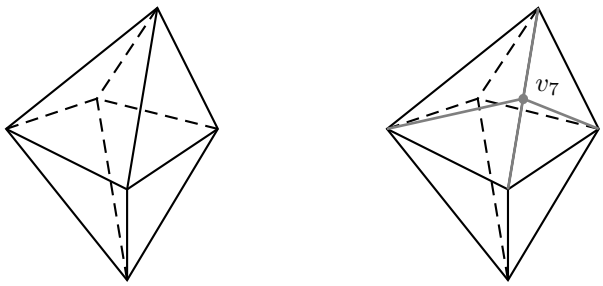


Figure 4.11: (a) triangulation for six vertices without vertices of degree three; (b) Adding a seventh vertex v_7 to the triangulation of six vertices

The triangulation of six vertices can also be considered as two stars of degree four, glued together. Therefore one new vertex can be added to two adjacent triangles of one star and the second new vertex can be added to the other star. It gives two triangulations of eight vertices (see figure 4.12).

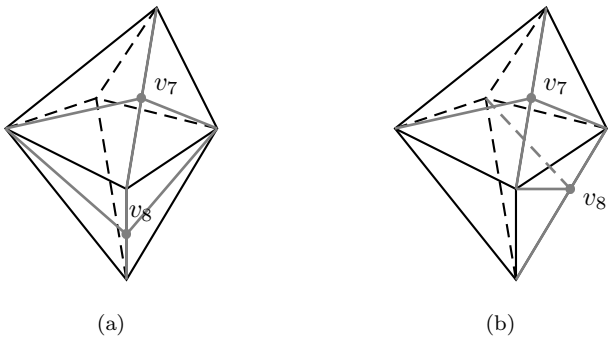


Figure 4.12: Triangulations for eight vertices: (a) with maximum cross-section six and with six vertices of degree four; (b) with four vertices of degree four

The triangulation in figure 4.12(a) is constructed by inserting two new vertices into adjacent edges of two stars, and the triangulation in figure 4.12(b) by inserting two new vertices into non–adjacent edges of two stars. The first triangulation has six vertices of degree four which are connected and the second one has four vertices of degree four, where two vertices of degree four are connected but are not connected with the other two vertices of degree four. Thus we define

the notion of a domain:

Definition 4.3.1 *A domain $dom(k)$ is a quadrilateral $u_1u_2u_3u_4$ ($deg(u_i) > 4$, $i = 1,..,4$) with k vertices $v_1,..,v_k$ of degree four inside and with the following internal connections: the vertices u_2 and u_4 are connected with vertices $v_1,..,v_k$ and the vertices u_1 and u_3 are only connected with vertices v_1 and v_k respectively (see figure 4.13). The number k of a domain $dom(k)$ is called cardinality.*

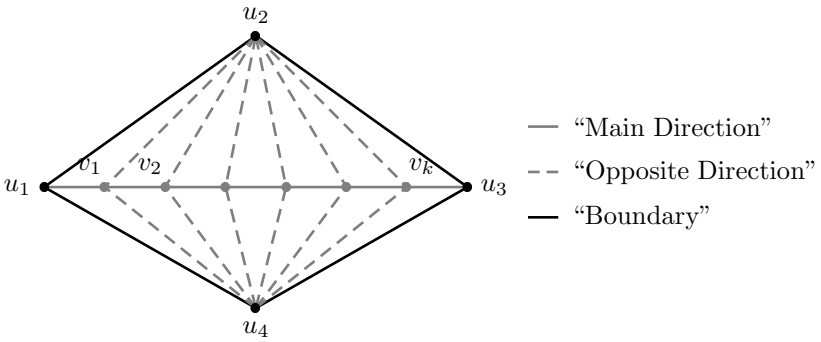


Figure 4.13: A domain $dom(k)$ with the cardinality k

Thus, for every triangulation we can determine several separate domains denoted as $dom_i(k_i)$, where $i = 1..l$, l is the number of domains of a triangulation and k_i is the cardinality of the i–th domain. In accordance with our definition we can say that the triangulations of six and seven vertices have one domain $dom(4)$ and $dom(5)$ respectively. One triangulation of eight vertices has one domain $dom(6)$ and the other one has two domains $dom_1(2)$ and $dom_2(2)$. By analogy with the method for dissectible polyhedra, we add l new vertices to l domains for constructing non-isomorphic triangulations of $N + l$ vertices.

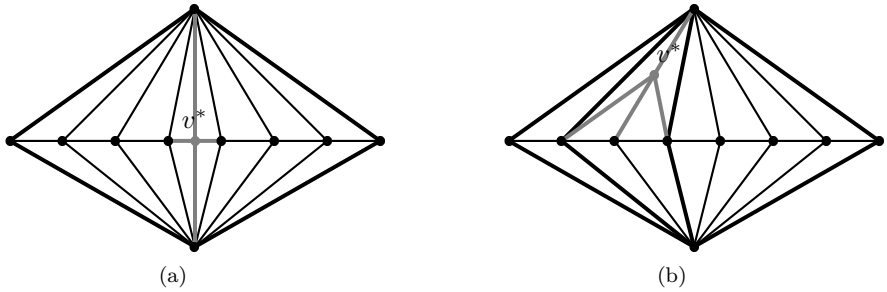


Figure 4.14: Inserting a new vertex v^* in (a) the main direction and in (b) an opposite direction of a domain

Now we have to define which edge inside a domain we have to consider for adding

a new point. $Dom(k)$ has $k+1$ edges which lie on the same line (it is possible to attain this by moving k vertices) and we call this line the "Main direction", all the other edges inside the domain are called the edges of the "Opposite direction" and four edges of quadrilateral $u_1u_2u_3u_4$ we call the "Boundary" (figure 4.13). It is easy to see that inserting a new vertex into any edge of the main direction increases the cardinality of the domain by one, but the number of domains of a triangulation remains the same (see figure 4.14(a)). Inserting a new vertex into an opposite direction divides the domain into two or three subdomains (it depends on which edge is chosen) and gives always one new subdomain $dom_1(2)$ and one or two new domains with less cardinality. In the example of figure 4.14(b) you can see that inserting a new vertex in the opposite direction of domain $dom(6)$ divides it into three subdomains $dom_1(0)$, $dom_2(2)$ and $dom_3(3)$ (from the left to the right). An domain with zero cardinality we call the *empty domain*. Thus, we define the addition of new vertices by the next rules:

1. We can insert a vertex in the main direction of a domain. This operation increases the cardinality of the domain by one, but does not change the number of domains;

2. We can insert several vertices simultaneously in a domain, for this we divide our domain into several subdomains with various cardinalities. Then we have to add m new vertices, where m is the number of subdomains. We insert then new vertices as follows:

   a) add a new vertex to the opposite direction if the cardinality of the subdomain is one;

   b) add a new vertex to the main direction if the cardinality of the subdomain is more than one.

For example, for a domain with cardinality three we can add vertices in the following way:

1. Add one vertex to the main direction of the domain (see figure 4.15(I));

2. Divide the domain into two subdomains $dom_1(1)$ and $dom_2(1)$ and add two vertices (see figure 4.15(II));

3. Divide the domain into one subdomain with cardinality one and two empty subdomains and add one (see figure 4.15(III)), two (see figure 4.15(IV') and (IV'')) and three (see figure 4.15(V)) vertices.

After adding m new vertices to a triangulation with l domains ($l \leq m$) we generate directly non–isomorphic triangulations of the second subset. In our

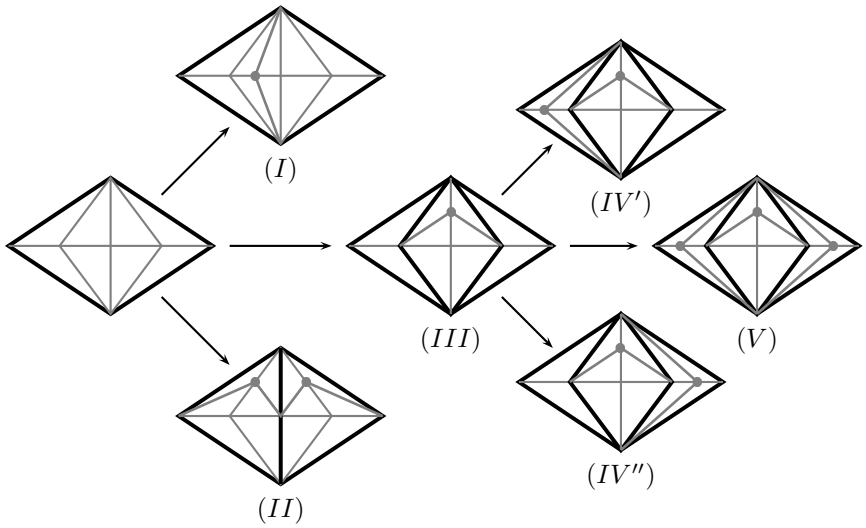


Figure 4.15: Dividing a domain into subdomains and adding l vertices

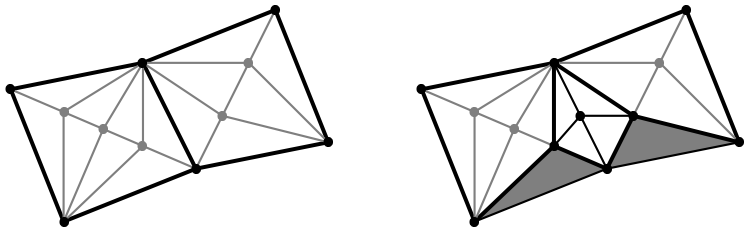


Figure 4.16: A triangulation (a) before and (b) after adding a new vertex into the boundary of two domains

construction we use only edges (quadrilaterals) which are inside a domain and do not use boundary edges or edges which are outside domains. Therefore we generate only a certain class of triangulations of the second subset. The remaining triangulations of the second subset we generate as well by adding new vertices into the boundary of two domains or an empty domain, but checking on isomorphisms is still needed.

More recently, we used a different idea to construct directly all non-isomorphic triangulations of the second subset, by adding a new vertex into a boundary of two domains or simultaneously adding a number of vertices into boundaries of domains. We reveal that adding a new vertex into the boundary of two domains (see figure 4.16), which have cardinalities greater than two, also produces directly non-isomorphic triangulations. Thus, we extend the class of non-

isomorphic triangulations which are directly constructed without isomorphisms, but we can not solved it completely. The main trouble is in adding a new vertex into the boundary of two domains with cardinalities less than three, because it produces domains with cardinality one or empty domains, for which it is very difficult to identify "Main" and "Opposite" direction. Thus, a triangulation which is constructed by adding a new vertex into the boundary of two domains with cardinality one or two, can be isomorphic to another one. The problem of adding new vertices into a boundary is not solved yet.

4.4 Conclusion

N	Dt^*_N	PDt^*_N	St^*_N	t^*_N
4	1	—	—	1
5	1	—	—	1
6	1	—	1	2
7	3	1	1	5
8	7	5	2	14
9	24	21	5	50
10	93	128	12	233
11	434	781	34	1249
12	2110	5355	130	7595
13	11002	38039	525	49566

Table 4.9: N is the number of vertices, $t^*_N = Dt^*_N + PDt^*_N + St^*_N$

In chapter 4 we developed methods for generating non–isomorphic triangulations (the result is presented in table 4.9). We divided all triangulations into two subsets. The first subset contains all triangulation with at least one vertex of degree three and the second one contains all triangulations with no vertex of degree three. The first subset of triangulations consists of two types of triangulations: dissectible and partly–dissectible triangulations. Partly–dissectible triangulations are dissectible triangulations which are generated from initial triangulations of the second subset. Thus, for generating them we firstly have

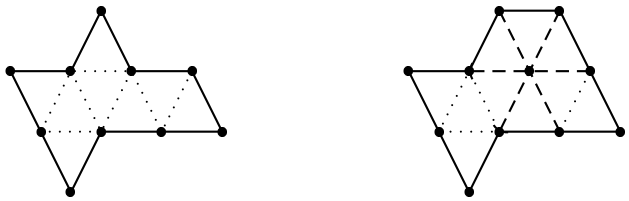
to construct triangulations of the second subset and then find their group of automorphisms.
For generating the triangulations of the first subset we developed a method which constructs directly non–isomorphic triangulations and does not require any checking on isomorphisms. This method only needs the knowledge of the group of automorphisms for an initial triangulation. The idea of the method is to add k new vertices per time to a triangulation in a special way, where $v_3 \leq k \leq 2(N-2)$.
For generating the triangulations of the second subset we firstly divide them into two types. The first type include all triangulations which are constructed by gluing a number of triangulations through one face. Generating the triangulations of the first type is a combinatorial problem and needs to define all different combinations of l triangulations (we did not consider that problem in this thesis). For generating triangulations of the second type we developed a method which is similar to the method for generating triangulations of the first subset. But in this case we consider two faces (quadrilateral) for adding a new point. For that we divide a triangulation into domains with internal vertices of degree four. Then new vertices are added to each domain for constructing directly non–isomorphic triangulations. The number of new vertices which is added per time equals the number of domains of a triangulation. The method for generating non–isomorphic triangulations of the second subset constructs some class of triangulations without extra checking on isomorphisms. Constructing the remaining triangulations of the second subset still needs extra checking on isomorphisms.

Open Problem 4.4.1 *How to avoid any checking on isomorphisms for generating triangulations of the second subset, which are constructed by adding a new vertex into the boundary of two domains.*

Chapter 5

Generating dissectible polygons (GDP)

In the previous chapter we presented the main algorithm for exhaustive generation of dissectible non-isomorphic triangulations, without checking on isomorphisms. An obvious idea is to apply the algorithm for generating a two–dimensional analogue of dissectible polyhedra, which are called dissectible polygons. In words, a polygon is called dissectible if it is possible to remove every vertex by cutting off one triangle, consecutively.



(a) (b)

Figure 5.1: (a) dissectible polygon; (b) non–dissectible polygon

For example, the polygon of figure 5.1(a) is dissectible but the polygon of figure 5.1(b) is not (in figure 5.1 we denote by dotted lines edges through which it is possible to cut off only one triangle and by dashed lines edges through which this is not possible). In section 4.1 we mentioned that a dissectible polyhedron can be presented as a 3–tree. Thus, a dissectible polygon can be viewed as a 2–tree which is embeddable in the plane. The problem of enumerating dissectible polygons with N vertices is a special case of enumerating labelled trees [MAT, MM63, Moo67], dissecting k-balls [BP71, BP72], enumeration of

rooted non-separable planar maps [BT64] and equivalent to the problem of generating all non–isomorphic (unlabelled) triangulations of a regular n-gon. A more difficult problem is the problem of enumerating unlabelled trees. A catalog of all of them up to 13 nodes was given by P.A. Morris [Mor]. R.C. Read in his study [Rea72] uses coding of an unlabelled tree in a special way in order to simplify checking on isomorphisms between two graphs.
Note, that for a dissectible polygon of N vertices the number of external edges is N, the number of internal edges is $N-3$ and the number of triangles is $N-2$. In this chapter we consider only the problem of generating dissectible polygons, which are presented as planar 2–trees. We give the analogue of the main algorithm 4.1.3 for generating dissectible polygons, which does not need any checking on isomorphisms. We also present two different approaches of the main algorithm for dissectible polygons. These two methods generate all non-isomorphic dissectible polygons as well but require less operations than the main method.

5.1 The main algorithm for *GDP*

In section 4.1 we described the main algorithm for generating non-isomorphic dissectible triangulations without checking on isomorphisms. The algorithm is based on adding simultaneously l new vertices of degree three to a triangulation, where $v_3 \leq l \leq 2(N-2)$, such that the first v_3 vertices from l must be added to one face of three faces of each star of degree three. The analogue of vertices of degree three for dissectible polygons are vertices of degree two. Therefore, for generating dissectible polygons we have to add l new vertices of degree two at every step, where $v_2 \leq l \leq N$ and v_2 is the number of vertices (stars) of degree two, such that the first v_2 vertices from l are added to one edge of two edges of each star of degree two. As a result, for the **2D** case algorithm 4.1.3 will be almost the same:

Algorithm 5.1.1 *The main algorithm for GDP.*

1. $P(3,1)$ *is a triangle,* $Dp^*_3 = 1$*;*
2. **for** $N := 3$ **to** $M-2$ **do**
   3. **for** $i := 1$ **to** Dp^*_N **do**
      begin
      4. *choose a polygon* $P(N, v_2)$ *with the following representation:*
         a) *All edges* $\{e_i\}$ $(i = 1, \ldots, N)$ *of* $P(N, v_2)$ *are written in the following order:*
         $\{e^1_{i+2(1-1)}\}^2_{i=1}, \ldots, \{e^{v_2}_{i+2(v_2-1)}\}^2_{i=1}, e_{2v_2+1}, \ldots, e_N$,
         where $\{e^j_{i+2(j-1)}\}^2_{i=1}$ *are two edges of the* j*–th star;*

b) Automorphism of groups $Aut(P(N, v_2))$.

5. $l := v_2$;

6. **repeat**

7. **if** $l = v_2$

then *Find all possible combinations of* v_2 *edges (each of them belongs to one star) in lexicographic order and write them into list* L_{v_2}. *The number of such combinations is* 2^{v_2};

else $L_l := L_{l-1}^{'} + \{e_i\}$, *where* $\{e_i\} := E_{external} \backslash \{L_{l-1}^{'}\}$, *i.e.,* $\{e_i\}$ *is the collection of the rest edges which have never been used in the list* $L_{l-1}^{'}$;

8. $L_l^{'} := reduct(L_l)$,

9. *Add* l *new vertices to* l *edges of a polygon* $P(N, v_2)$ *from the list* $L_l^{'}$ *and generate new non-isomorphic polygons* $P(N + l, l)$.

10. *Find* $Aut(P(N + l, l))$;

11. $l := l + 1$;

12. **until** $l > N$ *or* $N + l > M$;

end;

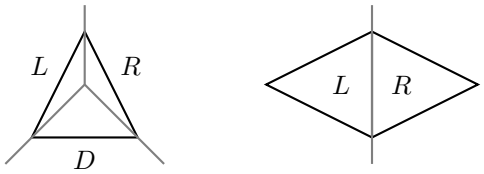
This algorithm for the **2D** case generates all non-isomorphic dissectible polygons. The result is presented in Table 5.1.

N	Dp_N^*	N	Dp_N^*
3	1	10	82
4	1	11	228
5	1	12	733
6	3	13	2282
7	4	14	7528
8	12	15	24834
9	27	16	83898

Table 5.1: Dp_N^* is the number of non–isomorphic dissectible polygons for N vertices

5.2 First approach for GDP (GDP_1)

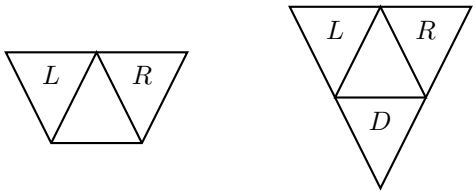
In this section we describe the first modified approach of the main method for dissectible polygons, which allows us to add at least two and at most $2k_{active}$ new vertices in a special way, where k_{active} is the number of vertices of degree two, which were added to a preceding polygon in a previous step. Because the initial triangulations[1] of three and four vertices have a symmetry of reflection and a symmetry of rotation an idea is to define "sides" of them and to label each "side" by different letters. For the triangulation of three vertices we define a right side, a left side and a inferior side, which are labelled by R, L and D, respectively (see figure 5.2(a)) and for the triangulation of four vertices we define a right and a left side with labels R and L (see figure 5.2(b)).



(a) (b)

Figure 5.2: The initial triangulations of three (a) and four (b) vertices

The triangulation of three vertices has three sides for adding new vertices. By the main algorithm we can add minimally two and maximally three new vertices to it. Adding two new vertices to any pair of sides gives isomorphic triangulations. So we add new vertices only to two sides simultaneously, for example, to the R–side and the L–side, which gives the triangulation of five vertices (see figure 5.3(a)). We also add three new vertices, which gives a triangulation of six vertices (see figure 5.3(b)).

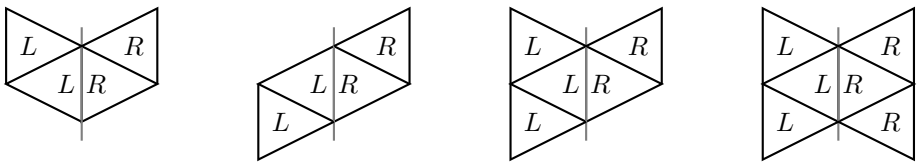


(a) (b)

Figure 5.3: Triangulations of five (a) and six (b) vertices

[1] in this chapter under triangulation we understand dissectible polygon.

The triangulation of four vertices has two sides and two active stars for adding new vertices. It means that we have to add two new vertices to stars of different sides. Because of automorphism of edges there are only two possibilities to do this (see figure 5.4(a) and 5.4(b)). We can also add three and four new vertices (see figure 5.4(c) and 5.4(d)).



(a) (b) (c) (d)

Figure 5.4: (a) and (b) are triangulations of six vertices; (c) is triangulation of seven vertices; (d) is triangulation of eight vertices

In the general case for adding new vertices we define two types of triangulations. The first type consists of all triangulations which have only active stars at two different sides, we call them "two–sides" triangulations, and the second type consists of all triangulations with active stars at three different sides, we call them "three–sides" triangulations. We denote the number of active stars of each side as follows: n_R is the number of active stars of degree two at the right side, n_L is the number of active stars of degree two at the left side and n_D is the number of active stars of degree two at the inferior side. New vertices have to be added only to the edges of an active star. Suppose we have a triangulation of N vertices with k active vertices of degree two. Adding new vertices then depends on the type of triangulation and proceeds along the next lines:

1. **"Two–sides" triangulation, it means $k = n_R + n_L$ or $k = n_R + n_D$ or $k = n_L + n_D$.**

   Suppose $k = n_R + n_L$. At the first step we choose all possible combinations of two stars such that they have different labels:

   $$comb2* = R_1L_1, \ldots, R_1L_{n_L}, \ldots, R_{n_R}L_1, \ldots, R_{n_R}L_{n_L}.$$

   (Combinations of two stars with the same label, for example R_iR_j or L_iL_j, are forbidden in $comb2*$). Then from all automorphic combinations we choose only non–automorphic combinations, reject all others and find $\widetilde{comb2*}$, which is a list of all non–automorphic combinations of the two stars. It is easy to see that the number of combination in the list $\widetilde{comb2*}$ is less or equal to the number of combinations $comb2*$ ($|\widetilde{comb2*}| \leq |comb2*|$). Every combination from $\widetilde{comb2*}$ has two stars. Adding two, three and four new vertices to each of them gives triangulations of $N+2$, $N+3$ and

$N+4$ vertices, respectively. Afterwards, we increase the number of stars for every combination of the list $\widetilde{comb2*}$ by one and find a new list $comb3*$, which after rejecting all automorphic combinations gives the list of all non–automorphic combinations of three stars $\widetilde{comb3*}$. To every combination from the list we add three, four, five and six new vertices that construct triangulations of $N+3$, $N+4$, $N+5$ and $N+6$ vertices, respectively, an so on. At the last step the list will have only one combination of all active stars to which it is possible to add at least k_{active} and at most $2k_{active}$ new vertices.

2. **"three–sides" triangulation, it means $k = n_R + n_L + n_D$.**

   In this case we choose three pairs of sides RL, RD, LD and take a non-automorphic pair of them, for which we use the same algorithm as for "two–sides" triangulations. Furthermore we also consider all three sides RLD and choose all possible combinations of three stars such that they have different labels:

   $$R_1L_1D_1, \dots R_1L_1D_{n_D}, \dots, R_{n_R}L_{n_L}D_1, \dots, R_{n_R}L_{n_L}D_{n_D}.$$

   (Combinations of two or three stars with the same label, for example $R_iR_jL_h$, are forbidden in the list). As in the case of "two–sides" triangulation we reject all automorphic combinations and find the list of all possible non–automorphic combinations of three stars. Every combination has three different labelled stars to which it is possible to add at least three and at most six new vertices for constructing triangulations of $N+3, \dots, N+6$ vertices. Then we increase each combination from the list by one star and perform the same operations. At the last step we have a list with only combinations of all active stars to which it is possible to add at least k_{active} and at most $2k_{active}$ new vertices.

The first approach of the main algorithm described above generates all possible non–isomorphic dissectible polygons without checking on isomorphisms by adding l new vertices simultaneously, where $2 \le l \le 2k_{active}$. This is done, such that, if we have a "two–sides" ("tree-sides") triangulation, the first two (three) vertices must be added to one edge of two (three) different stars and the two (three) stars must lie at different sides.

Theorem 5.2.1 *GDP_1 generates all possible non-isomorphic triangulations.*

Proof. All triangulations are constructed by adding new vertices to active stars of a triangulation. Moreover, at least two active stars must lie on two different sides (the proof of this fact is presented below). Therefore, if we represent a triangulation as a tree with triangles as nodes and an initial triangulation

(a) (b)

Figure 5.5: (a) one rooted node; (b) two rooted nodes

as rooted node, then such a tree has at least two leaves which lie at the same distance of the rooted node, and this distance is maximal. A triangulation constructed from an initial triangulation of three vertices has only one rooted node (see figure 5.5(a)) and a triangulation constructed from an initial triangulation of four vertices has two rooted nodes. One node is used for constructing left branches and the other is used for constructing right branches (see figure 5.5(b)). We prove the theorem in two steps:

- **First step: The algorithm generates all possible triangulations**. Suppose some triangulation is missing. We present this triangulation in a tree–like form (see figure 5.6) and find a path with maximum length for it.

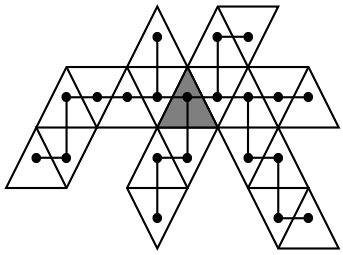


Figure 5.6: Representation in a tree-like form for a triangulation

There are two possibilities:

1. **Maximum length of the path is even,** $maxlength = 2n$**.**
   It means that the path has $2n$ edges and $2n + 1$ nodes. We take the middle node, $(n + 1)$, as a rooted node. Then we find all paths from the rooted node to every leaf of the tree. We can determine at least two paths with the same length n (see figure 5.7). After deleting all leaves from the branches of the length n we will get a new tree again with at least two equal paths, but of length $n - 1$. Deleting at every step all leaves of length $n-1, n-2, n-3, \ldots, 1$ gives at the end one node which is the rooted node. It means that this triangulation is

constructed from the initial triangulation of three vertices, and for such a triangulation there are at least two leaves which are situated on different sides. We have a contradiction with our assumption.

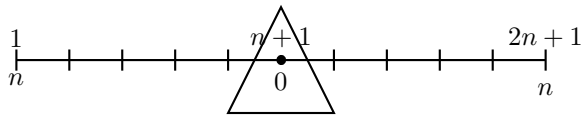


Figure 5.7: Maximum length of path is even

2. **Maximum length of the path is odd,** $maxlength = 2n + 1$**.** It means that the path has $2n + 1$ edges and $2n + 2$ nodes. We take as the middle edge, $(n+1)$, edge which connects the rooted node n of the left side and the rooted node $n+1$ of the right side. Then we find all paths from the rooted nodes to every leaf of the tree by choosing these rooted nodes. We can determine at least two paths with the same length n (see figure 5.8). Like in the previous case, deleting at every step all leaves of length $n, n-1, n-2, n-3, \ldots, 1$ gives two nodes which are the rooted nodes. It means that the triangulation is constructed from the initial triangulation of four vertices. We obtain again a contradiction with our assumption.

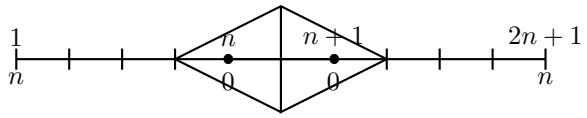


Figure 5.8: Maximum length of path is odd

Now we show that two leaves always lie on different sides of the rooted nodes n and $n + 1$. Suppose two leaves lie on the same side (it does not matter on which one). They have the same rooted node, and the path length from the rooted node to each leaf is n. Hence, the maximum path length will be $n + n = 2n$. It means that the triangulation is constructed from the initial triangulation of three vertices and that it has two leaves on the different sides.

- **Second step: All triangulations are non–isomorphic**

  1. Triangulations which are constructed from different initial triangulations are non–isomorphic because they have different numbers of

rooted nodes.

2. Triangulations which are constructed from the same initial triangulation are non–isomorphic. All triangulations are constructed from an initial triangulation of three or four vertices. Suppose two triangulations are isomorphic. It means that there is a one-to-one correspondence between vertices and their connections. The number of active vertices is also equal. After cutting off all active vertices of each triangulation new triangulations are isomorphic and the number of active vertices are equal. We continue cutting off till we arrive at the initial triangulation or at two equal triangulations. The last is possible only if we took an automorphic combination of active stars, but this contradicts our construction.□

Algorithm 5.1.1 takes the following form:

Algorithm 5.2.2 GDP_1

1. $P(3,1)$ *is a triangle,* $Dp_3^* = 1$*;*

2. **for** $N := 2$ **to** 2 **do**

3. **for** $i := 1$ **to** Dp_N^* **do**
begin

4. choose a polygon $P(N, v_2^a, dir)$*, where* v_2^a *is the number of active stars of degree two and dir is the number of directions, with the following representation:*

a) All edges $\{e_i\}$ $(i = 1, \ldots, 2v_2^a)$ *of* $P(N, v_2^a, dir)$ *are written in the following order:*

$$\{e^1_{i+2(1-1)}\}^2_{i=1}, \ldots, \{e^{v_2^a}_{i+2(v_2-1)}\}^2_{i=1},$$

where $\{e^j_{i+2(j-1)}\}^2_{i=1}$ *are two edges of the j–th star;*

b) Automorphism of groups $Aut(P(N, v_2^a, dir))$*.*

5. $l := dir$*;*

6. **repeat**

7. **if** $l = dir$

then *Find all possible combinations of l edges (each of them belongs to stars of l different directions) in lexicographic order and write them into list* $L_{v_2^a}$*.*

else $L_l := L'_{l-1} + \{e_i\}$*, where* $\{e_i\} := E_{v_2^a} \backslash \{L'_{l-1}\}$*,*
i.e., $\{e_i\}$ *is the collection of the rest edges which have never been used in the list* L'_{l-1} *and belong to active stars of* $P(N, v_2^a, dir)$*;*

8. $L_l^{'} := reduct(L_l)$,
9. *Add* l *new vertices to* l *edges of a polygon* $P(N, v_2^a, dir)$ *from the list* $L_l^{'}$ *and generate new non-isomorphic polygons* $P(N+l, l, dir)$.
10. *Find* $Aut(P(N+l, l, dir))$;
11. $l := l + 1$;
12. **until** $l > 2v_2^a$ *or* $N + l > M$;

end;

5.3 Second approach for GDP (GDP_2)

In the previous section the first approach of the main algorithm for dissectible polygons was explained. Using this algorithm we can divide all dissectible polygons into two types. The first type contains all triangulations that are constructed from the initial triangulation of three vertices. The second type consists of all triangulations that are constructed from the initial triangulation of four vertices. In this section we present an improvement of the algorithm for the first type of triangulations. The purpose of this improvement is to reduce the number of operations.
In the previous section it was proved that every triangulation has at least two paths of the maximum length and both lie on two different sides. For the first type of triangulations there are three combinations for choosing the two sides. They are RL, RD and LD. Using rotation of triangulations along middle vertices of the initial triangulation of three vertices, every side can be transformed to each other, *i.e.*,

$$RL \rightarrow LD \rightarrow DR \text{ or } RL \rightarrow RD \rightarrow DL.$$

Let us define by the "potential edge" an edge which is used for construction of the inferior direction branch and consider the first steps of the algorithm. The initial triangulation is the triangulation of three vertices. To this triangulation we add two new vertices in the right and left directions and add three new vertices to all three directions. New triangulations will be triangulations for five (see figure 5.3(a)) and six (see figure 5.3(b)) vertices respectively. At the next step to triangulation of five vertices we can add two, three and four new vertices to both directions R and L and construct new triangulations for seven, eight and nine vertices which have only two directions and a "potential edge". Also to triangulations of five vertices we can add three, four and five new vertices such that new triangulations will have all three directions and will not have a "potential edge". Triangulations of six vertices have no "potential edge" and have three directions, therefore we must add at least three vertices to all directions, and so on.

In the general case the idea of the second approach is to fix only one pair of sides for adding new vertices to the initial triangulation of three vertices, for example, the sides RL. Then there are three possibilities for adding new vertices (define by $R(i)$, $L(i)$ and $D(i)$ directions R, L and D, respectively, with path length i from the initial triangulation):

1. **A triangulation has two directions $R(n)$, $L(n)$ and a "potential edge".**
   New l vertices are added only to two directions R and L simultaneously ($2 \leq l \leq 2k_{active}$). Then new triangulations will also have two directions $R(n+1)$ and $L(n+1)$ (see figure 5.9).

2. **A triangulation has two directions $R(n)$, $L(n)$ and a "potential edge".**
   New l vertices are added to two directions R and L simultaneously, like in the previous case, and one new vertex is added to the "potential edge" ($3 \leq l \leq 2k_{active}+1$). Then new triangulations will have three directions $R(n+1)$, $L(n+1)$ and $D(1)$ (see figure 5.9).

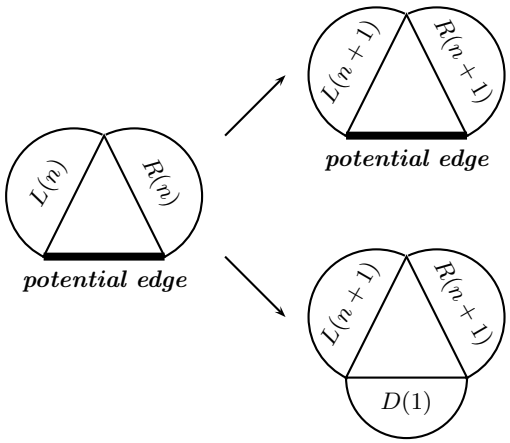


Figure 5.9:

3. **A triangulation has three directions $R(n)$, $L(n)$ and $D(k)$.**
   New l vertices are added in three directions R, L and D simultaneously ($3 \leq l \leq 2k_{active}$). Then new triangulations will have three directions $R(n+1)$, $L(n+1)$ and $D(k+1)$ (see figure 5.10).

As can be seen this algorithm does not need make a choice of two directions from three, which reduces the number of operations.

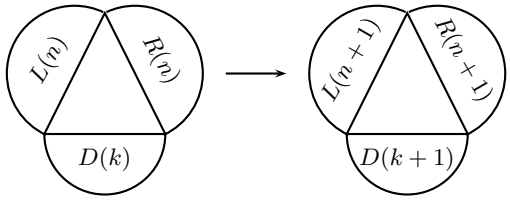


Figure 5.10:

Theorem 5.3.1 *GDP_2 generates all possible non–isomorphic triangulations.*

PROOF. It is easy to prove that two triangulations which have different path lengths from the initial triangulation will be non–isomorphic. Let the triangulations T_1 and T_2 have $R_1(n)$, $L_1(n)$ and $D_1(k_1)$ and $R_2(n)$, $L_2(n)$ and $D_2(k_2)$ respectively. There are two possibilities:

1. $k_1 \neq k_2$ It is easy to see that the triangulations T_1 and T_2 are non-isomorphic.

2. $k_1 = k_2 = k$ Suppose the triangulations T_1 and T_2 are isomorphic. If we cut off all leaves of the length n for the L-side and R-side and all leaves of the length k_i for the D–side of the triangulation T_i $(i = 1, 2)$. New triangulations are isomorphic. We proceed with cutting off the vertices till we get triangulations which are equal. It might be only a triangulation with $R(n-l)$, $L(n-l)$ and $D(k-l)$ $(l \leq k)$, or the initial triangulation. It means that T_1 and T_2 are constructed from the same triangulation, but by the algorithm this is not possible. We get a contradiction with our assumption.

The proof that we do not miss any triangulation is similar to the proof in previous section.□

5.4 Conclusion

In chapter 5 we adapted the main algorithm for dissectible polyhedra to the two–dimensional case, *i.e.*, for dissectible polygons. We also made two approaches for the main algorithm which reduces the number of operations in constructing dissectible polygons. The algorithms require adding only $2 \leq \tilde{l} \leq 2k$ new vertices per time to a polygon, whereas, in the main algorithm $k \leq l \leq N$. It seems that these two approaches can be applied for dissectible polyhedra as well, but we did not implement them yet.

Open Problem 5.4.1 *Extend two approaches of this chapter for constructing dissectible triangulations in* **3D**.

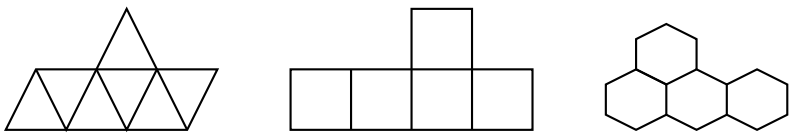
Chapter 6

Triangular Animals

In this chapter we adapt our algorithm for generating dissectible triangulations (namely GDP_1) to solve an important combinatorial problem of generating tree–like triangular animals without checking on isomorphism. This problem stays somewhat apart from the main stream of our thesis, however it gives a nice practical application of our theoretical results.

6.1 Applying GDP_1 for triangular animals

The combinatorial problem known as the cell-growth problem has a long history and was included in the list of unsolved problems in the enumeration of graphs by F. Harary in 1960 [Har60]. An analogue of this problem is the enumeration of so-called *animals*, where an *animal* is the collection of cells of equal shape starting from a single one, which grows step by step in the plane by adding at each step a cell in such a way that the new cell has an only connection with a side of an already presented cell. The cells of animals are always triangles,



(a) (b) (c)

Figure 6.1: (a) triangular, (b) square and (c) hexagonal animals

squares or hexagons because these are the only regular polygons which can fill the plane. Correspondingly, animals are called triangular animals, square animals or hexagonal animals (figure 6.1).

At present, this problem has received much interest in mathematics, physics, biology and computer science [Har68, KMST85, Tri92, WW86].
There are some papers, in which different types of animals are enumerated, but it seems that no one can generate all of them for a large number of cells (20, 30, 40 and etc.) without checking on isomorphisms. Recent work was done by E.V. Konstantinova. She considers triangular [Kon00b] and square [Kon00a] animals and generates all of them up to 13 and 11 cells respectively, but her algorithm still performs checking on isomorphisms [Kon01].

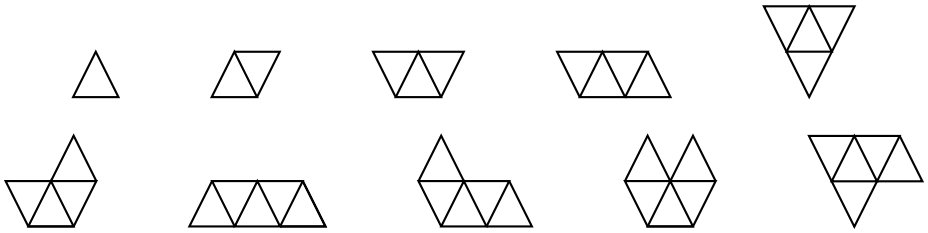


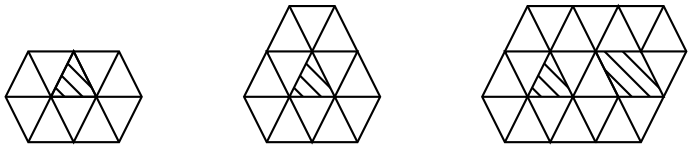
Figure 6.2: "Triangular animals"

In this chapter we deal with triangular animals. All of them, up to five cells, are shown in figure 6.2. All triangular animals are divided in two different classes $\mathcal{S}(\mathbb{R}^3)$ and $\mathcal{M}(\mathbb{R}^3)$ [Kon00b]. The class $\mathcal{S}(\mathbb{R}^3)$ consists of *simply-connected* animals embedded in $\mathbb{R}^3$, *i.e.,* all animals which have only one external boundary. The class $\mathcal{M}(\mathbb{R}^3)$ consists of *multiply-connected* animals embedded in $\mathbb{R}^3$, *i.e.,* all animals with two or more boundaries. One boundary is external and the others are internal, also called "holes".

(a) (b)

Figure 6.3: *simply-connected* triangular animals

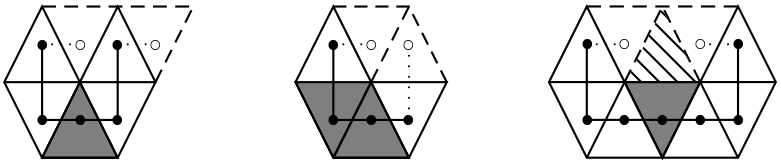
We divide all simply-connected triangular animals into two subclasses. The first subclass contains all animals without vertices inside (figure 6.3(a)) which are also called *tree-like* triangular animals (*TlTrAn*). The second subclass contains the rest, namely, all animals with at least one vertices inside (figure 6.3(b)).
All multiply-connected triangular animals we also divide into subclasses, in this case three. The first subclass contains animals with the property that every internal boundary has a common vertex with the external boundary (fig-



(a) (b) (c)

Figure 6.4: *Multiply-connected* triangular animals

ure 6.4(a)). Such animals are also called pseudo-connected triangular animals (*PcTrAn*). The second subclass consists of animals with the property that external and internal boundaries have no common vertices (figure 6.4(b)). All the other animals in $\mathcal{M}(\mathbb{R}^3)$, which have both properties, belong to the third subclass and are called "mixed" (figure 6.4(c)).



(a) (b) (c)

Figure 6.5: Triangulation which are constructed without applying global restrictions

In the previous chapter the main method for generating all non-isomorphic dissectible polygons (method *GDP*) has been considered, and two of its approaches (GDP_1 and GDP_2). Dissectible polygons as well as triangular animals consist of triangles. Therefore, if we consider all cells of a dissectible polygon as equilateral cells of equal size, then the problem of generating *TlTrAn* becomes a particular case of the problem of generating dissectible polygons. As a result any algorithm for dissectible polygons can be used for constructing *TlTrAn* and the most appropriate candidate is GDP_1. However, after applying this algorithm directly, some triangular animals with vertices inside (figure 6.3(b)) and triangular animals of class $\mathcal{M}(\mathbb{R}^3)$ (figure 6.4) will be constructed. For these types of triangular animals, the method GDP_1 does not guarantee a non-isomorphic construction. Hence, some extra restrictions have to be applied to it in order to avoid constructing such animals. Two global restrictions, which appear at first glance, can be defined as follows:

1. The set of new vertices should not overlap with the set of vertices of the preceding triangular animals. Thus, triangular animals with vertices inside are not constructed, for example, such an animal as in figure 6.5(a);

2. The set of new vertices should contain only different elements. Thus, triangular animals of the second subclass of $\mathcal{S}(\mathbb{R}^3)$ and of class $\mathcal{M}(\mathbb{R}^3)$ are not constructed, for example such animals as in figure 6.5(b) and figure 6.5(c).

After applying these restrictions to algorithm GDP_1, the algorithm for generating *TlTrAn* will follow ($TrAn_N^*$ is the number of non-isomorphic tree-like triangular animals of N vertices):

Algorithm 6.1.1 *The algorithm for generating TlTrAn.*

1. $TlTrAn(3, 1, \mathbf{dir})$ *is triangle,* $TrAn_3^* = 1$ *(*$\mathbf{dir} = 2, 3$*).*
$TlTrAn(4, 2, 2)$ *is two triangles glued via one edge,* $TrAn_4^* = 1$*;*

2. **for** $N := 3$ **to** $M - 2$ **do**

3. **for** $i := 1$ **to** $TrAn_N^*$ **do**
begin

4. choose an animal $TlTrAn(N, v_2^a, \mathbf{dir})$*, where* v_2^a *is the number of active stars of degree two and* **dir** *is the number of directions, with the next representation:*

a) Edges $\{e_i\}_{i=1}^{2v_2^a}$ *of* $TlTrAn(N, v_2^a, \mathbf{dir})$ *belong to active stars and are written in the next order:*
$\{e^1_{i+2(1-1)}\}_{i=1}^2, \ldots, \{e^{v_2^a}_{i+2(v_2^a-1)}\}_{i=1}^2$*, where* $\{e^j_{i+2(j-1)}\}_{i=1}^2$ *is two edge of* j*–th active star;*

b) Automorphism of edges $\{e_i\}_{i=1}^{2v_2^a}$ $Aut(TlTrAn(N, v_2^a, \mathbf{dir}))$*;*

c) $\mathbf{V} = \{v_i\}_{i=1}^N$ *is the set of all vertices of the preceding TlTrAn;*

5. $l := \mathbf{dir}$*;*

6. **repeat**

7. **if** $l = \mathbf{dir}$

then *Find all possible combination of* l *edges (each of them belongs to stars of* l *different directions) in lexicographic order and write them into list* L_l*;*

else $L_l := L_{l-1}^{'} + \{e_i\}$*, where* $\{e_i\} := \{e_j\}_{j=1}^{2v_2^a} \backslash \{L_{l-1}^{'}\}$*,*
i.e., $\{e_i\}$ *is the rest edges which have never been used in the list* $L_{l-1}^{'}$ *and belong to active stars of* $TlTrAn(N, v_2^a, \mathbf{dir})$*;*

8. $L_l^{'} := reduct(L_l)$*;*

8′. Find the set of new vertices $\mathbf{newV_1}$ *which have to be added to edges of* $L_l^{'}$*;*

8″. $L_l^{''} := delcomb_1(L_l^{'})$ *delete all combinations for which* $\mathbf{V} \cap \mathbf{newV_1} \neq \varnothing$ *(global restriction 1);*

$8'''$. $L_l''' := delcomb_2(L_l'')$ *deleting all combinations for which* $v_i = v_j$ $((i \neq j) = 1..l)$, $v_i \in \mathbf{newV_1}$ *(global restriction 2);*

9. *Add* l *new vertices to* l *edges of the animal* $TlTrAn(N, v_2^a, \mathbf{dir})$ *from the list* L_l''' *and generate new non-isomorphic animals* $TlTrAn(N+l, l, \mathbf{dir})$.

10. *Find* $Aut(TlTrAn(N+l, l, \mathbf{dir}))$;

11. $l := l + 1$;

12. **until** $l > 2v_2^a$ *or* $N + l > M$;

end.

Now let us take a tree-like triangular animal and look at the edges of its active stars. The purpose is to find local restrictions for reducing the number of operations in steps $8' - 8'''$. For that we consider three cases:

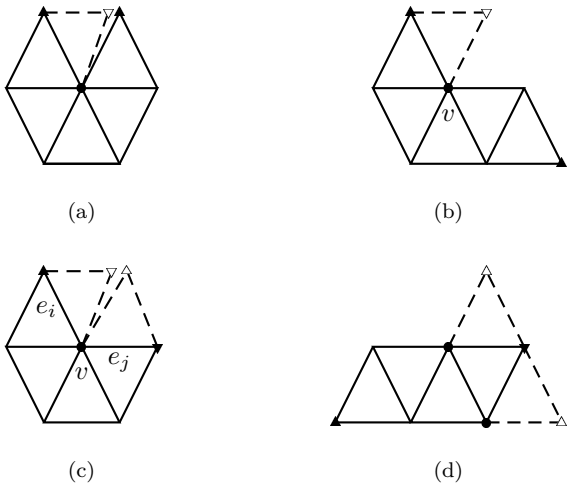


Figure 6.6: Searching local restrictions in construction *TlTrAn*

- Assume an edge of an active star has a vertex of degree six. Then adding a new vertex increases its degree to seven (see figure 6.6(d)), which is forbidden. As a result all edges of a preceding triangular animal which have a vertex of degree six have to be deleted from the list of edges for adding new vertices. Thus, in step *4a*) all edges with a vertex of degree six are not considered;

- Suppose now an edge of an active star has a vertex v of degree five. There are two possibilities:

  - The active star which contains that edge has no adjacent star in vertex v (see figure 6.6(b)). As a result a new vertex can be added to the edge because the degree of v increases to six;
  - Two active stars are adjacent in the vertex v (see figure 6.6(c)). Assume edges e_i and e_j belong to two adjacent stars and v is their common vertex. Adding two new vertices simultaneously to e_i and e_j increases the degree of vertex v to seven, which is again forbidden. But adding a new vertex to e_i or e_j increases the degree of v to six, which is allowed. As a result, the combination, which has two adjacent edges with common vertex of degree five, has to be deleted from the list of edges for adding new vertices. Thus, at step 7 L_l has no such combination, and edge $\{e_i\}$ is not considered for adding a new vertex if it has a common vertex with an edge of L'_{l-1};

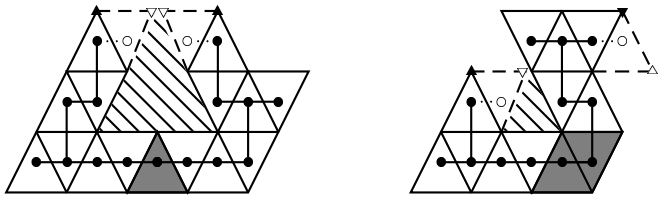
- Now consider an edge of an active star which have vertices of degree less that five. New vertices can be always added in such a case.

Consequently, we can formulate two local restrictions which have to be imposed in the first step of algorithm:

1. All edges of active stars, which have a vertex of degree six, do not have to be considered in the list of edges for adding new vertices;

2. Combinations, which have two adjacent edges with a common vertex of degree five, do not have to be considered in the list of edges for adding new vertices.

Finally, the algorithm with two global and two local restrictions generates all non-isomorphic tree-like triangular animals and avoids the construction of animals different from *TlTrAn*. The result is presented in table 6.1.



(a) (b)

Figure 6.7: Triangulations are constructed by using only two local restrictions

It is essential to use both global and local restrictions. Because the algorithm with only local restrictions will generate all tree-like triangular animals as well as

N	$TlTrAn^*_N$	N	$TlTrAn^*_N$
3	1	12	405
4	1	13	1041
5	1	14	2825
6	3	15	7541
7	4	16	20525
8	11	17	55633
9	23	18	152181
10	62	19	416188
11	148	20	1143526

Table 6.1: $TlTrAn^*_N$ is the number of non–isomorphic triangular animals for N vertices

multiply-connected triangular animals or triangulations with self–intersections. For example, in figure 6.7(a) and figure 6.7(b) you can see the result of using only two local restrictions for constructing *TlTrAn*. Both triangulations have no vertices of degree more that six but after increasing the number of cells new vertices can connect the boundary of triangulation, that is avoided by only global restrictions. Thus, in order to observe local restrictions we need to check that after increasing *TlTrAn* by cells an obtained animal has no vertices of degree more than six, whereas by imposing global restrictions we avoid self–intersections in a constructed animal (*i.e.*, overlapping its cells).

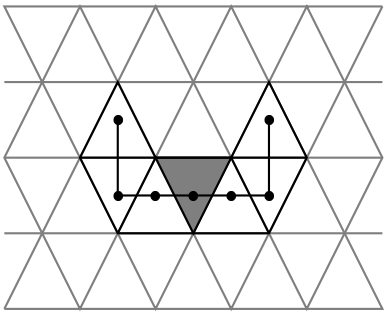


Figure 6.8: Triangular coordinate system

As we mentioned in the beginning of this section the plane can be filled by triangles. In the procedure of constructing *TlTrAn* we consider equilateral triangles. Below we give the notion of boundary of *TlTrAn*. For that we introduce a special triangular coordinate system in the plane, three axes of which are three lines that pass through the sides of equilateral triangle (see figure 6.8). As a result, any vertex in the plane is defined by three coordinates (x_1, x_2, x_3), that indicate the indexes of the lines that intersect in this vertex, with respect to the axes. This gives the possibility to easily identify whether two vertices lie on the same line, or how far one vertex is shifted along the coordinate lines with respect to another vertex.

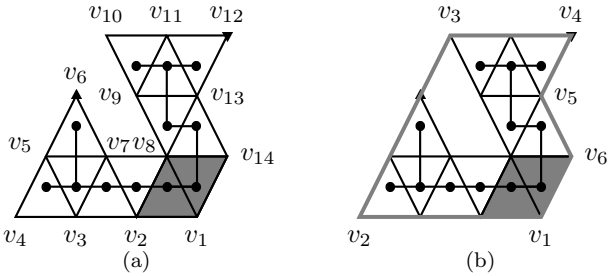


Figure 6.9: Identifying new boundary for a triangulation

Using the above described triangular coordinate system we introduce a new concept of boundary for *TlTrAn*. As an example, we consider triangulation of figure 6.9. Triangulation of figure 6.9(a) has the boundary $v_1v_2...v_{14}$. Firstly, note, that we don't need to keep the vertices v_2, v_3, v_5, v_9 and v_{11} with degree four in the list of the boundary vertices because the edges adjacent to it belong to the same line; and it easy to define whether a new added vertex belongs to a straight segment of the boundary, for example, to the segment v_1v_4, or not (global restrictions). Secondly, note, that vertices of degree six are also not needed in the boundary. Thus, vertex v_8 has to be deleted from the boundary and the new boundary will be $v_1v_4v_6v_7v_9v_{10}v_{12}v_{13}v_{14}$. But for the new boundary the degree of vertex v_7 becomes six and we can again delete it. Finally, we simplify the boundary of a triangulation in such a way that it has no vertices of degree four and six. In figure 6.9(b) you can see the new boundary (bold gray) with only six vertices instead of fourteen for the triangulation of figure 6.9(a).

6.2 Conclusion

In chapter 6 we considered an application of the algorithm GDP_1 to the cell-growth problem for generating Tree-like Triangular Animals. We present an algorithmic construction that complies to two local and two global restrictions.

We also define the boundary for $TlTrAn$ in a special way that allows us to reduce the amount of the boundary vertices, and, as a result, reduce the number of checks on self–connections in constructed animals.

Part II

Constructing triangulations interpolating real data

Chapter 7

Basic notions and definitions

7.1 Some concepts of differential geometry[1]

The differential geometry of curves and surfaces has two principal aspects. One can be called classical and the other one, global. Classical differential geometry is the study of local properties of curves and surfaces. By local properties one means those properties which depend only on the behavior of the curve or surface in the neighborhood of a point. The methods which are used for studying such properties are methods of differential calculus. Therefore, one defines curves and surfaces, which are considered in differential geometry, by functions, which can be differentiated a certain number of times. Global differential geometry studies the influence of the local properties on the behavior of the entire curve or surface, *i.e.*, it deals with the relations between local and global (topological) properties. Below we give some notions of classical differential geometry for which the main interest lies in regular curves and surfaces [dC76].
By a curve or surface one characterizes certain subsets of $\mathbb{R}^3$ that are, in a certain sense, one–dimensional or two–dimensional respectively and to which the methods of differential calculus can be applied[2]. Therefore, such subsets can be defined through differentiable functions. A real function of a real variable is differentiable (or smooth) of order n if it has derivatives of order n at all points. Such a function belongs to class $\mathbb{C}^n$. If for some function $n = \infty$, then such a

[1] Most of the definitions and theorems of this section were taken from the book of M.P. do Carmo "Differential Geometry of Curves and Surface". We refer the reader to the original text for more details.

[2] $\mathbb{R}^3$ is the set of triple (x,y,z) of real numbers; in general case it should be $\mathbb{R}^d$ but in this thesis only $\mathbb{R}^3$ is considered.

function is called infinity differentiable and it belongs to class $\mathbb{C}^\infty$. A function of class $\mathbb{C}^0$ is called continuous.

Definition 7.1.1 *A parameterized differentiable curve is a differentiable map $\alpha : I \to \mathbb{R}^3$ of order n of an open interval $I = (a, b)$ of the real line $\mathbb{R}$ into $\mathbb{R}^3$.*

The word *differentiable* in this definition means that α is a correspondence which maps each $t \in I$ into a point $\alpha(t) = (x(t), y(t), z(t)) \in \mathbb{R}^3$ in such a way that the functions $x(t)$, $y(t)$ and $z(t)$ are differentiable of order n. The variable t is called the *parameter* of the curve. The word *interval* is taken in a generalized sense, *i.e.*, it does not exclude the cases $a = -\infty$ and $b = +\infty$. By $x'(t)$, $y'(t)$ and $z'(t)$ one denotes the first derivative of x, y and z at the point t respectively. The vector $\alpha'(t) = (x'(t), y'(t), z'(t)) \in \mathbb{R}^3$ is called the *tangent vector* (or *velocity vector*) of the curve α at t.
Let $\alpha : I \to \mathbb{R}^3$ be a parameterized differentiable curve. Then for each $t \in I$ where $\alpha'(t) \neq 0$ there is a well-defined straight line, which contains the point $\alpha(t)$ and the tangent vector $\alpha'(t)$, which is called the *tangent line* to α at t. The existence of such a tangent line at every point of a curve is an essential condition for the study of local differential geometry. Therefore, any point where $\alpha'(t) = 0$ is called a *singular point* of α.

Definition 7.1.2 *A parameterized differentiable curve $\alpha : I \to \mathbb{R}^3$ is called regular if $\alpha'(t) \neq 0$ for all $t \in I$.*

By means of tangent vector $\alpha'(t)$ to a regular parameterized curve $\alpha : I \to \mathbb{R}^3$, $t \in I$ one can define the *arc length* of $\alpha(t)$ from the point t_0

$$s(t) = \int_{t_0}^{t} \parallel \alpha'(t) \parallel dt.$$

We can also define the angle between two curves at an intersection point t as the angle between two corresponding tangent vectors.
The analogue of a regular curve is a regular surface which is obtained by taking pieces of a plane, deforming them and arranging them in such a way that the resulting object has no sharp points, edges or self–intersections. We give the definition of a regular surface [dC76]:

Definition 7.1.3 *A subset $\mathrm{S} \subset \mathbb{R}^3$ is a regular surface if, for each $\mathrm{p} \in \mathrm{S}$, there exists a neighborhood V in $\mathbb{R}^3$ and a map $\mathbf{x} : \mathrm{U} \to \mathrm{V} \cap \mathrm{S}$ of an open set $\mathrm{U} \subset \mathbb{R}^2$ onto $\mathrm{V} \cap \mathrm{S} \subset \mathbb{R}^3$ such that*

1. *$\mathbf{x}$ is differentiable. This means that if we write*

$$\mathbf{x}(\mathrm{u}, \mathrm{v}) = (\mathrm{x}(\mathrm{u}, \mathrm{v}), \mathrm{y}(\mathrm{u}, \mathrm{v}), \mathrm{z}(\mathrm{u}, \mathrm{v})), \qquad (\mathrm{u}, \mathrm{v}) \in \mathrm{U},$$

the functions $\mathrm{x}(\mathrm{u}, \mathrm{v})$, $\mathrm{y}(\mathrm{u}, \mathrm{v})$ and $\mathrm{z}(\mathrm{u}, \mathrm{v})$ have continuous partial derivatives of all orders in U;

2. $\mathbf{x}$ *is a homeomorphism. Since* $\mathbf{x}$ *is continuous by condition 1, this means that* $\mathbf{x}$ *has an inverse* $\mathbf{x}^{-1} : \mathrm{V} \cap \mathrm{S} \to \mathrm{U}$ *which is continuous; that is,* $\mathbf{x}^{-1}$ *is the restriction of a continuous map* $\mathrm{F} : \mathrm{W} \subset \mathbb{R}^3 \to \mathbb{R}^2$ *defined on an open set* W *containing* $\mathrm{V} \cap \mathrm{S}$*;*

3. *(The regularity condition). For each* $\mathrm{q} \in \mathrm{U}$*, the differential* $\mathrm{d}\mathbf{x}_\mathrm{q} : \mathbb{R}^2 \to \mathbb{R}^3$ *is one–to–one.*

The mapping $\mathbf{x}$ is called a parametrization or a system of (local) coordinates in (a neighborhood of) p. The neighborhood $V \cap S$ of p in S is called a coordinate neighborhood [dC76]. From the definition it follows that for a regular surface it is possible to use the method of differential calculus, there are no self–intersections and there exists a set of tangent vectors at a point $p \in S$ which constitutes a plane.

Proposition 7.1.4 *Let* $\mathbf{x} : \mathrm{U} \subset \mathbb{R}^2 \to \mathrm{S}$ *be a parametrization of a regular surface* S *and let* $\mathrm{q} \in \mathrm{U}$*. The vector subspace of dimension* 2*,*

$$\mathrm{d}\mathbf{x}_\mathrm{q}(\mathbb{R}^2) \subset \mathbb{R}^3$$

coincides with the set of tangent vectors to S *at* $\mathbf{x}(\mathrm{q})$*.*

By the proposition the plane $d\mathbf{x}_q(\mathbb{R}^2)$, which passes through $\mathbf{x}(q) = p$, does not depend on the parametrization $\mathbf{x}$. This plane is called *tangent plane* to S at p and is denoted by $T_p(S)$. The tangent plane also allows to define the angle of two intersecting surfaces at a point of intersection [dC76].
In differential geometry there are two forms to define a regular surface, which are called *fundamental forms*. The fundamental forms are extremely important and useful in determining the metric properties of a surface.
First, we define the *First Fundamental Form* via the inner product on $T_p(S) \subset \mathbb{R}^3$, where $T_p(S)$ is the tangent plane at the point p of the regular surface S. If $w_1, w_2 \in T_p(S)$, then $< w_1, w_2 >_p$ is the inner product of w_1 and w_2 as vectors in $\mathbb{R}^3$. To this inner product, which is a symmetric bilinear form, there corresponds a quadratic form $\mathbf{I}_p : T_p(S) \to \mathbb{R}$ given by [dC76]:

$$\mathbf{I}_p(w) =< w, w >_p = |w|^2 \geq 0.$$

Definition 7.1.5 *The quadratic form* $\mathbf{I}_\mathrm{p}$ *on* $\mathrm{T_p(S)}$ *is called the first fundamental form of the regular surface* $\mathrm{S} \subset \mathbb{R}^3$ *at* $\mathrm{p} \in \mathrm{S}$*.*

Geometrically, the first fundamental form allows us to measure on the surface lengths of curves, angles of tangent vectors, area of regions without referring back to the ambient space $\mathbb{R}^3$ in which the surface lies.
The first fundamental form can be expressed in the basis $\{\mathbf{x}_u, \mathbf{x}_v\}$ associated to a parametrization $\mathbf{x}(u, v)$ at p. Because a tangent vector $w \in T_p(S)$ is the

tangent vector to a parameterized curve $\alpha(t) = \mathbf{x}(u(t), v(t))$, $t \in (-\epsilon, \epsilon)$, with $p = \alpha(0) = \mathbf{x}(u_0, v_0)$, one obtain

$$\mathbf{I}_p(\alpha'(0)) = E(u')^2 + 2Fu'v' + G(v')^2,$$

where the value of the functions are computed for $t = 0$, and

$$E(u_0, v_0) =< \mathbf{x}_u, \mathbf{x}_u >_p,\ F(u_0, v_0) =< \mathbf{x}_u, \mathbf{x}_v >_p,\ G(u_0, v_0) =< \mathbf{x}_v, \mathbf{x}_v >_p$$

are the coefficients of the first fundamental form in the basis $\{\mathbf{x}_u, \mathbf{x}_v\}$ of $T_p(S)$. In a coordinate neighborhood of $\mathbf{x}(u, v)$, the functions $E(u, v)$, $F(u, v)$, $G(u, v)$ are differentiable [dC76].
Thus, the arc length s of a parameterized curve $\alpha : \mathbf{I} \to S$, given by $s(t) = \int_0^t \sqrt{\mathbf{I}(\alpha'(t))}dt$, can be computed by

$$s(t) = \int_0^t \sqrt{E(u')^2 + 2Fu'v' + G(v')^2}dt.$$

We can also compute the area of a bounded region of a regular surface. A (regular) domain of S is an open and connected subset of S such that its boundary is the image of a circle by a differentiable homeomorphism which is regular except at a finite number of points. A region of S is the union of a domain with its boundary. A region of $S \subset \mathbb{R}^3$ is *bounded* if it is contained in some ball of $\mathbb{R}^3$. The area of a bounded region is defined in the following definition [dC76]:

Definition 7.1.6 *Let* $\mathbb{R} \subset$ S *be a bounded region of a regular surface contained in the coordinate neighborhood of the parametrization* $\mathbf{x}$: U $\subset \mathbb{R}^2 \to$ S. *The positive number*

$$\iint_{\mathrm{Q}} |\mathbf{x}_u \wedge \mathbf{x}_v| \mathrm{dudv} = \mathrm{A(R)}, \qquad \mathrm{Q} = \mathbf{x}^{-1}(\mathrm{R}),$$

is called the area of R.

It is convenient to observe that the integrand of $A(R)$ can be rewritten as $|\mathbf{x}_u \wedge \mathbf{x}_v| = \sqrt{EG - F^2}$.
Now we introduce the notion of orientation of a surface. Intuitively speaking, the choice of orientation of the tangent plane $T_p(S)$ of a regular surface S at an arbitrary point p prompts to an orientation in a neighborhood of p, that is a positive movement along a sufficiently small closed curve about each point of the neighborhood. Thus, if it is possible to choose the orientation for every $p \in S$ so that in the intersection of any two neighborhoods the orientations coincide, then S is called *orientable*. Otherwise, S is called *nonorientable*. Orientation is a global property of a surface, in the sense that it involves the whole surface.

No we can give a classification of closed (compact and connected) surfaces. Let A be a subset of $\mathbb{R}^3$. We say that $p \in \mathbb{R}^3$ is a *limit point* of A if every neighborhood of p in $\mathbb{R}^3$ contains a point of A distinct from p. If A contains all its limit points then A is *closed*. If A is contained in some ball of $\mathbb{R}^3$ then A is *bounded*. A is called a *compact set* if it is closed and bounded. A compact surface has a triangulation with a finite number of triangles. Therefore, in this thesis we only deal with closed surfaces.
Another important geometric concept, to which the notion of tangent plane leads, is the curvature of a regular surface. Below we introduce the definitions of the Gauss map, principal curvatures, Gaussian and mean curvatures.
Let us define the notion of orientation in more technical terms. Suppose $\mathbf{x} : U \subset \mathbb{R}^2 \to S$ is a parametrization of a regular surface S at a point p. A unit normal vector at each point of $\mathbf{x}(U)$ can be chosen by the rule

$$N(q) = \frac{\mathbf{x}_u \wedge \mathbf{x}_v}{|\mathbf{x}_u \wedge \mathbf{x}_v|}(q), \qquad q \in \mathbf{x}(U).$$

Thus, if $V \subset S$ is an open set in S and $N : V \to \mathbb{R}^3$ is a differentiable map which associates to each $q \in V$ a unit normal vector at q, we say that N *is a differentiable field of unit normal vectors on* V. We shall say that a regular surface is orientable if it admits a differentiable field of unit normal vectors defined on the whole surface; the choice of such a field N is called an orientation of S. Now we can give the definition of the Gauss map [dC76]:

Definition 7.1.7 *Let* $\mathrm{S} \subset \mathbb{R}^3$ *be a surface with an orientation* N. *The map* $\mathrm{N} : \mathrm{S} \to \mathrm{S}^2$ *takes its values on the unit sphere*

$$\mathrm{S}^2 = \{(x, y, z) \in \mathbb{R}^3 | x^2 + y^2 + z^2 = 1\}.$$

The map $\mathrm{N} : \mathrm{S} \to \mathrm{S}^2$*, thus defined, is called the Gauss map of* S.

The Gauss map is differentiable and its image $N(S) = \{q \in S^2 : q = N(p)\}$ is called the *spherical image* of the oriented surface S. The differential $dN_p : T_p(S) \to T_p(S)$ of the Gauss map allows us to introduce the *Second Fundamental Form* [dC76]:

Definition 7.1.8 *The quadratic form* $\mathbf{II}_\mathrm{p}$*, defined in* $\mathrm{T_p(S)}$ *by* $\mathbf{II}_\mathrm{p} = - < \mathrm{dN_p(v), v} >$ *is called the second fundamental form of* S *at* p.

Using the Gauss map we can define the quantity $K(p)$, known as the *Gaussian curvature* of S at p:

$$|K(p)| = \lim_{U(p) \downarrow p} \frac{Area(N(U(p)))}{Area(N(p))},$$

where the limit is taken as the neighborhood $U(p)$ contracts down to the point p. There is another interpretation of the Gaussian curvature in terms of *principal curvatures* [dC76]:

Definition 7.1.9 *Let* C *be a regular curve in* S *passing through* $\mathrm{p} \in \mathrm{S}$, k *is the curvature of* C *at* p, *and* $\cos\theta =< \mathrm{n}, \mathrm{N} >$, *where* n *is the normal vector to* C *and* N *is the vector to* S *at* p. *The number* $\mathrm{k_n} = \mathrm{k}\cos\theta$ *is then called the normal curvature of* $\mathrm{C} \subset \mathrm{S}$ *at* p.

Definition 7.1.10 *The maximum normal curvature* k_1 *and the minimum normal curvature* k_2 *are called the principal curvatures at* p.

Definition 7.1.11 *Let* $\mathrm{p} \in \mathrm{S}$ *and let* $\mathrm{dN_p} : \mathrm{N_p(S)} \to \mathrm{T_p(S)}$ *be the differential of the Gauss map. The determinant of* $\mathrm{dN_p}$ *is the Gaussian curvature* K *of* S *at* p. *The negative of half of the trace of* $\mathrm{dN_p}$ *is called the mean curvature* H *of* S *at* p.

Thus, from the last definition we can see that the Gaussian curvature K and the mean curvature H in terms of the principal curvatures can be written as follows:

- The Gaussian curvature K of S at p is the product of the principal curvatures
$$K = k_1 k_2;$$
- The mean curvature H of S at p is the average of the principal curvatures
$$H = \frac{k_1 + k_2}{2}.$$

In other words, the Gaussian curvature is a measure how rapidly a surface S pulls away from the tangent plane $T_p(S)$ in a neighborhood of a point $p \in S$. The mean curvature of a surface S is a function $H : S \to \mathbb{R}$ which measures how much S is "bent".

Definition 7.1.12 *A point* p *of a surface* S *is called*

1. *Elliptic if* $det(\mathrm{dN_p}) > 0$;
2. *Hyperbolic if* $det(\mathrm{dN_p}) < 0$;
3. *Parabolic if* $det(\mathrm{dN_p}) = 0$, *with* $\mathrm{dN_p} \neq 0$;
4. *Planar if* $\mathrm{dN_p} = 0$.

From the definition follows that the Gaussian curvature indicates whether a region of a surface is elliptic, hyperbolic or parabolic. Namely, at an elliptic point the Gaussian curvature is positive, *i.e.,* both principal curvatures have the same sign. At a hyperbolic point the Gaussian curvature is negative, *i.e.,* the principal curvatures have the opposite signs. At a parabolic point the Gaussian curvature is zero, but one of the principal curvatures is not zero. Finally, at a planar point, all principal curvatures are zero [dC76].
The mean curvature indicates whether a region is "full" or "hollow". The signs of mean and Gaussian curvatures yield eight "basic" surface region types (in terms of terrain modelling [PJB86]):

$H > 0$, $K > 0$ "peak" surface region (convex);

$H = 0$, $K = 0$ flat surface region;

$H < 0$, $K > 0$ "pit" surface region (concave);

$H = 0$, $K < 0$ minimal surface region;

$H > 0$, $K = 0$ "ridge" surface region;

$H > 0$, $K < 0$ "saddle ridge" surface region;

$H < 0$, $K = 0$ "valley" surface region;

$H < 0$, $K < 0$ "saddle valley" surface region.

With this information about curvatures we proceed with the discrete analogue of mean and Gaussian curvatures.

7.2 Discrete mean and Gaussian curvatures

The authors of many publications that deal with representation and visualization of an object, base their work on spline interpolation of the surface of an object. The simplest spline interpolation works well in **2D** space for representing a curve by a piecewise linear interpolant through the points of the curve without local self–intersection. In **3D** the simplest spline of a surface is given by a triangular mesh. Because surfaces in $\mathbb{R}^3$ are well–studied by differential geometry it seems logical that using knowledge of differential geometry might help to give a better understanding of triangular meshes and it might help to construct good representations of objects. However, local differential geometry is not suited to describe curvatures of triangulated polyhedral surfaces, because neighborhoods of most of the points of these surfaces represent a plane domain. On the other hand, the methods of global differential geometry should be extended in order to treat non–regular features of polyhedral surfaces. The theoretical methods

investigating non–regular geometry from the position of (global) differential geometry have been developed by the Russian mathematician A.D. Aleksandrov in his theory of non–regular surfaces [AZ67]. In this thesis we restrict ourselves to the basic part of their theory, mainly, to the theory of manifolds with polyhedral metrics.

Definition 7.2.1 *A space with polyhedral metrics is a special space of a two–dimensional manifold of bounded curvature in which each point has a neighborhood isometric to the lateral surface of a cone*[3]*.*

Polyhedral metrics adopts discrete analogies of concepts of classical differential geometry which are easy to compute directly from a given data set. One of the first works to adapt notions of the theory of non–regular surfaces for applications was done by L. Alboul and R. van Damme in 1994 [AvD95a].
In this work we want to generate a triangulation from a given data set of points, which has to represent the surface of an object. Knowledge of differential geometry gives full information about any surface and was presented in the previous section. In case of a triangulation we have a triangular mesh for which characteristics of differential geometry can not be properly defined along edges and vertices because it is not differentiable. If we consider a triangulation as a piecewise linear approximation of some surface we can try to combine the knowledge of differential geometry with information that is given by the triangular mesh. In particular, we want to consider the Gauss curvature K and mean curvature H at vertices of a triangulation which for smooth objects are defined in terms of the principal curvatures k_1 and k_2 by $K = k_1 k_1$ and $H = \frac{k_1+k_2}{2}$ respectively.

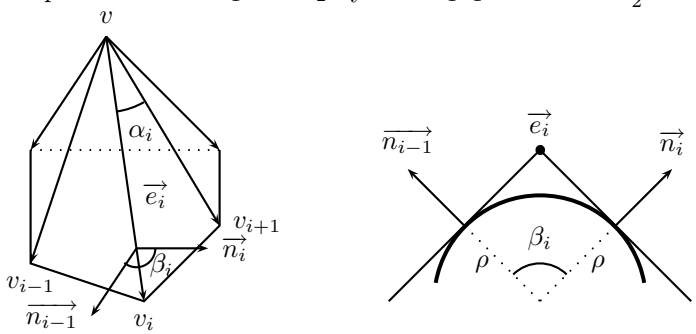


Figure 7.1: Star of vertex v

In a series of papers [AvD95a, AvD95b, AvD96, DHKL01, MD02] many authors considered applications of two curvatures K and H to the procedure of the

[3]From the definition follows that a ball $U(a, r)$ in polyhedral metrics, when r is sufficiently small, is isometric to the plane circle of radius r with the center a, or to a surface of the convex cone with the apex a and the cone generator of length r. The full angle $\theta(a)$ at point a is the value $\theta(a) = \frac{S(r)}{r}$, where $S(r)$ is the length of boundary of a ball $U(a, r)$ and r is sufficiently small. Points a are called vertices of a polyhedral metrics and $\theta(a) \neq 2\pi$.

optimization of a triangulation. In order to compute the curvatures, let us consider a vertex v of some triangulation and its star $Str(v)$ (see definition 2.2.2). $Str(v)$ consists of n ordered vertices v_i and edges $\overrightarrow{e_i} = v_i - v$ $(i = 1..n)$. The angle between two successive edges with the end point v is $\alpha_i = \angle(\overrightarrow{e_i}, \overrightarrow{e_{i+1}})$ and the dihedral angle at an edge $\overrightarrow{e_i}$ between the normals of two adjacent triangles of $Str(v)$ is $\beta_i = \angle(\overrightarrow{n_{i-1}}, \overrightarrow{n_i})$, where $n_i = \frac{\overrightarrow{e_i} \times \overrightarrow{e_{i+1}}}{\| \overrightarrow{e_i} \times \overrightarrow{e_{i+1}} \|}$ [4] (see figure 7.1). As a result, the expressions of the analogues of the integral Gaussian curvature around a vertex v and the integral mean curvature along an edge e are given respectively by

$$K(v) = 2\pi - \sum_{i=1}^{n} \alpha_i \quad \text{and} \quad H(e) = \beta_i \parallel \overrightarrow{e_i} \parallel . \tag{7.2.1}$$

The first formula is the simplest version of the Gauss–Bonnet theorem [dC76]. The second formula can be illustrated by the following consideration: let us replace each edge by a small cylinder of radius ρ, that joins the adjacent faces tangentially (see figure 7.1), in order to obtain a smooth surface on which K and H are integrable functions. Then the methods of differential geometry can be applied and after some manipulations the formulas (7.2.1) can be obtained [AvD95a, dC76].

7.3 Criteria for optimizing the triangulation

In chapter 1 it was mentioned that for constructing an optimal triangulation of a data set, one can use the edge flipping algorithm of Lawson in combination with some optimality criterion. Below we briefly present three well-known optimality criteria for constructing an optimal triangulation, and in the next section we give two recently obtained criteria: minimizing the analogue of the absolute Gaussian curvature, which is well-known as the Tight criterion, and minimizing the analogue of the absolute mean curvature, which are defined for polyhedral metrics [AvD95a, AvD95b, DHKL01].

- **Minimizing the area of the resulting object [O'R87].**
  Let f_i be a face with three vertices v_i^1, v_i^2 and v_i^3. Then the area of f_i is

$$S_{f_i} = \frac{1}{2} \parallel \overrightarrow{a} \parallel \cdot \parallel \overrightarrow{b} \parallel \sin(\gamma),$$

  where $\overrightarrow{a} = v_i^2 - v_i^1$, $\overrightarrow{b} = v_i^3 - v_i^1$ and $\gamma = \frac{\overrightarrow{a} \times \overrightarrow{b}}{\| \overrightarrow{a} \| \cdot \| \overrightarrow{b} \|}$.
  The sum of all face areas f_i for a triangulation T_k is

[4] index 0 is identified with n and index $n+1$ with 1.

$$S(T_k) = \sum_i S_{f_i}.$$

As a result, the optimal triangulation T with smallest surface area is defined by

$$S_{min}(T) = min_k\{S(T_k)\}.$$

- **Minimizing the angle between normals (ABN) [DLR90].**
  Let t_i and t_j be two faces with common vertices v_{ij}^1 and v_{ij}^2 and v_i and v_j belong to t_i and t_j respectively. Then the angle between the two normals to t_i and t_j is

$$\alpha_{ij} = \frac{\arccos(\overrightarrow{n_i} \cdot \overrightarrow{n_j})}{\| \overrightarrow{n_i} \| \cdot \| \overrightarrow{n_j} \|},$$

  where $n_s = \frac{\overrightarrow{a_s} \times \overrightarrow{b_s}}{\|\overrightarrow{a_s} \times \overrightarrow{b_s}\|}$, $a_s = v_{ij}^1 - v_s$ and $b_s = v_{ij}^2 - v_s$ $(s = i, j)$.
  The sum of all angles between normals α_{ij} for a triangulation T_k is

$$ABN(T_k) = \sum_{i,j} \alpha_{ij}.$$

  As a result, the optimal triangulation T with minimum sum of angles between normals is defined by

$$ABN(T) = min_k\{ABN(T_k)\}.$$

- **Minimizing a certain energy functional [QS90].**
  Let the energy measure be defined by

$$\varepsilon(s, \triangle) = \sum_{\{T_j\}} \int_{T_j} [(\frac{\partial^2}{\partial x^2} s(x,y))^2 + 2(\frac{\partial^2}{\partial x \partial y} s(x,y))^2 + (\frac{\partial^2}{\partial y^2} s(x,y))^2] dxdy,$$

  where $\{T_j\}$ denotes the set of triangles of the triangulation $\triangle$. This expression represents the energy of the so-called "thin plate", which is approximately equal to $\int_A (k_1^2 + k_2^2) dA$, where k_1 and k_2 are the principal curvatures of the surface. The expression $(k_1^2 + k_2^2)$ can be rewritten as follows

$$k_1^2 + k_2^2 = (k_1 + k_2)^2 - 2k_1k_2 = 4H^2 - 2K.$$

Therefore, one gets

$$\int_A (k_1^2 + k_2^2) dA = 4 \int_A H^2 dA - 2 \int_A K dA.$$

So, one needs to minimize the left side of this equation that leads to minimization of the expression which stands in the right side of the same equation. However, from the Gauss-Bonnet theorem follows that the last integral is a constant. It means that minimization of $\int_A (k_1^2 + k_2^2) dA$ yields minimization of the expression $\int_A H^2 dA$ and vice versa. $\int_A H^2 dA$ is called the Willmore energy. Recently A. Bobenko [Bob04] introduced a discrete analogue of this curvature.

7.4 Minimizing abs. mean and abs. Gauss curvatures

In section 7.2 we show that the notion of discrete curvatures is strictly related to the concept of an angle.
The discrete analogue of Gauss curvature in a vertex v, defined by angles of plane triangles incident at v is

$$K(v) = 2\pi - \sum_{i=1}^{n} \alpha_i.$$

L. Alboul and R. van Damme in their study [AvD95b] used three types of curvatures for a triangulation considered as a polyhedral surface. More theoretical descriptions of these curvatures can be found in the works of Burago, Banhoff and Kühnel [Ban67, BK97, Bur68].

- The positive curvature $K^+(v)$.
  Suppose that through the vertex v there passes some (local) supporting plane of a triangulation. Then this vertex lies on the boundary of convex hull of $Star(v)$. We denote the star of v in the boundary of this convex hull by $Star^+(v)$ and will call it the star of convex cone of a vertex. The curvature $K^+(v)$ of $Star^+(v)$ is called the positive curvature of v. If there is no supporting plane through v then we put $K^+(v)$ equal to zero;

- The negative curvature $K^-(v)$.
  $K^-(v) = K^+(v) - K(v)$;

- The absolute curvature $\hat{K}(v)$.
  $\hat{K}(v) = K^+(v) + K^-(v)$.

Based on the above described notions the following sets of vertices are identified:

1. A vertex v is called a proper convex vertex if the following statements hold:
   - $K^+(v) = K(v) > 0$ and $K^-(v) = 0$;
   - the Gauss map is non–intersecting and positively oriented;
   - any plane spanned by two subsequent edges is a supporting plane.
2. A vertex v is called a proper saddle vertex if:
   - $K^-(v) = -K(v) > 0$ and $K^+(v) = 0$;
   - the Gauss map might have multiple loops all negatively oriented;
   - there exists locally no supporting plane.
3. A vertex v is called a mixed vertex if:
   - $K^-(v) > 0$ and $K^+(v) = 0$;
   - the Gauss map is intersecting;
   - the vertex v admits locally a supporting plane.

Examples of these three types of vertices are given in figure 7.2.

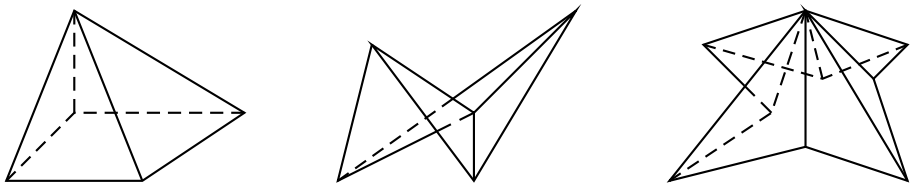


Figure 7.2: Three types of vertices: proper convex, proper saddle and mixed

Thus, the total absolute curvature $|K|$ of a triangulation was given by the following expression:

$$|K| = \sum_{v_\alpha \text{ convex}} K^+(v_\alpha) + \sum_{v_\beta \text{ saddle}} K^-(v_\beta) + \sum_{v_\gamma \text{ mixed}} (K^+(v_\gamma) + K^-(v_\gamma)).$$

The discrete analogue of mean curvature on a strip along an edge e is:

$$H = \beta_i \parallel \overrightarrow{e_i} \parallel .$$

Thus, the total absolute mean curvature was given by

$$|H| = \sum_e |H(e)|.$$

In section 7.1 it was mentioned that the Gaussian curvature indicates whether a region is elliptic, hyperbolic or parabolic, whereas the mean curvatures indicates whether a region "full" or "hollow". These two curvatures are associated with a surface and describe the shape of a surface sufficiently good. Therefore evaluation of curvatures always was important in surface reconstruction in order to extract the shape characteristics of the approximated surface. In general, the curvatures are computed on a given model, or sometimes directly from the data by using methods of finite differences and later finite elements. The first attempt where the discrete curvatures were straightforward used for characterizing a polyhedral model of a piece of terrain was work of [FS91]. Let us also note that the absolute extrinsic Gaussian curvature and the discrete mean curvature were applied for optimizing a polyhedral surface by a smooth one [BK79]. The optimization methods based on curvature optimality criteria, where also introduced already some time ago. Most of these methods were based on discretized forms of classical curvatures. In 1994 a new methods that directly based on a discrete analogue of total absolute Gaussian curvature was introduced [AvD95a, AvD95b]. The optimality criterion based on the discrete form of Mean curvature was also then announced. Later on both methods were explored in more details [AER04, AHA02, AKTvD99, Alb03, AvD95a, AvD95b, AvD97, AvD02, DHKL01, MD02]. In this work we continue to investigate both criteria in order to obtain a good initial triangulation.

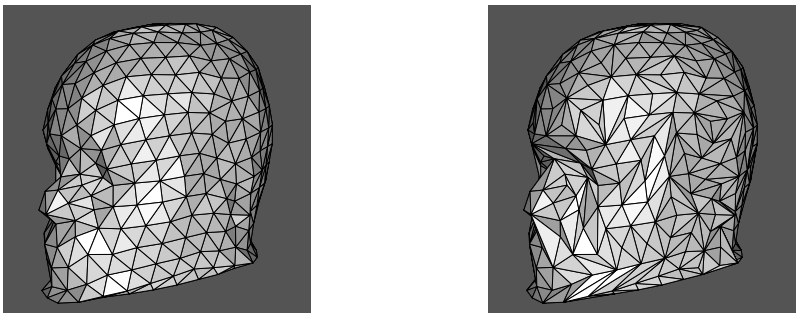


Figure 7.3: Before (left) and after (right) applying the Tight criterion

Motivation, for example, to use the criterion of minimizing the discrete analogue of the total absolute Gaussian curvature is related to the concept of Tight submanifolds [BK97, Kui70]. One of the main properties of two–dimensional tight submanifolds in $\mathbb{R}^3$ is that they possess the minimal total absolute curvature, or, equivalently, the height function on a tight surface will have the minimum number of non-degenerate critical points that a height function can have on a closed surface. Therefore, a tight surface of genus 0 is a convex surface. In practice we deal with the data which is in general non-convex. However, we

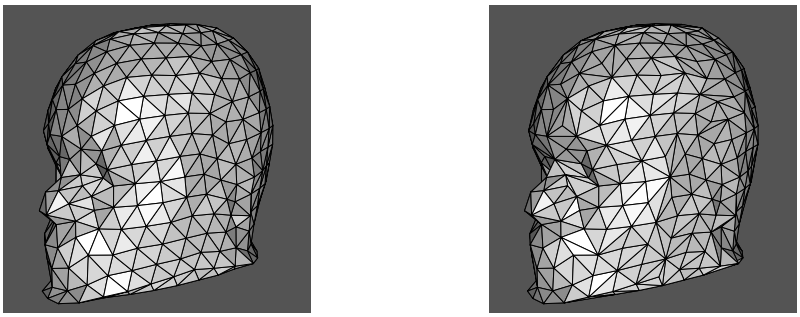


Figure 7.4: Before (left) and after (right) applying the mean criterion

can assume that the data are taken in such a way, that all important features (creases, curvatures, etc.) of a surface can be extracted from the data, and that the representation of a surface as a triangular mesh does not add features not present in the data. Therefore, the properties of Tight submanifolds make the idea to construct a triangulation of minimum total absolute curvature very appealing. As we already said above, surface triangulations of the minimum total absolute extrinsic curvature, (**MTAEC**), were introduced some time ago [AvD95a, AvD95b]. The MTAEC triangulation coincides with the convex triangulation of the data and gives the global optimum if the data are in convex position, but it has some undesirable result for non–convex data (see figure 7.3). The properties of triangulations with MTAEC for non-convex data are still mostly unknown (see [AvD96] for a review).

Triangulations of minimum mean curvature (the mean criterion) were also considered [AER04, AKTvD99, DHKL01], and experimental results seem very promising (see figure 7.4), but it also known very little about their properties. The mean criterion also gives a good result if we take as initial triangulation the triangulation produced by the Tight criterion (see figure 7.3 (right)). The result is presented in figure 7.5.

In both cases (see figure 7.4 and figure 7.5) the mean criterion was applied to the object with a "good" initial representation. But an initial triangulation can be also very bad. As an example we consider the initial triangulation on the left of figure 7.6. After applying the mean criterion a new triangulation will differ from the triangulation on the right of figure 7.4 or 7.5. Thus, application of the same optimality criterion to two different initial triangulations of the same data set can produce different optimal triangulations. In this case it is clear that the mean criterion can not generate the same triangulations by using edge flipping algorithm (EFA). Indeed, the global optimum of any criterion cannot be found in general. The global minimum might also be not unique, moreover, it might be far away from the input configuration [FMS03]. A local minimum, which is

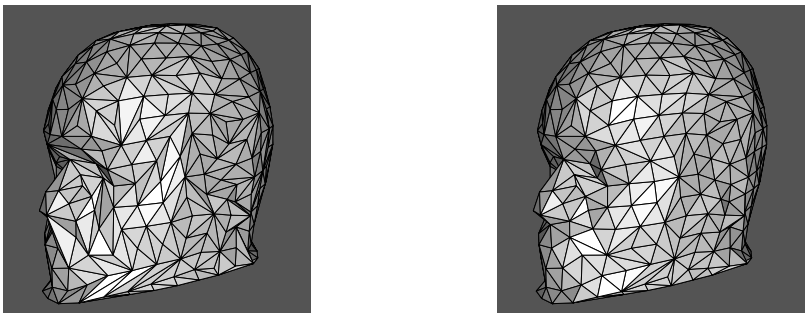


Figure 7.5: Before (left) and after (right) applying the mean criterion

as close as possible to the initial configuration, might be more meaningful.
In the next chapter we consider the multi–edge flipping algorithm (MEFA) which with the mean criterion gives a "good" optimal triangulation for a "bad" initial one and consider a method for constructing an initial triangulation from a scattered data set.

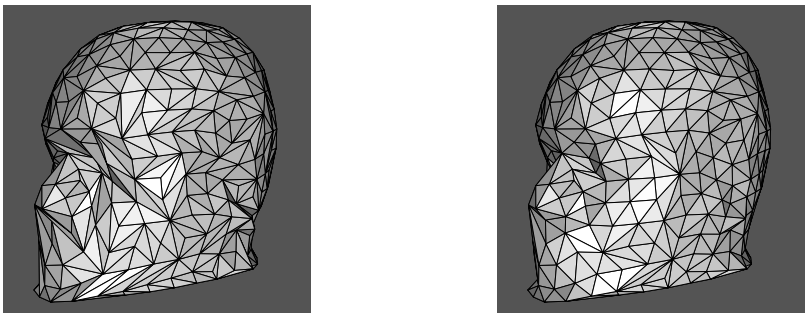


Figure 7.6: Bad initial triangulation (left) and triangulation after applying the mean criterion with EFA (right)

Chapter 8

Triangulation

8.1 Edge flipping and Multi–Edge flipping algorithms

In the previous chapter we gave basic notions and definitions of Differential Geometry. We also described discrete mean curvature, Gaussian curvature and a few well–known methods of optimizing a triangulation. In this chapter we describe MEFA and show what kind of advantage and disadvantage it has over EFA.
Suppose an initial triangulation was constructed. Then a new task is to optimize it via a sequence of transformations such that it will be a good representation of the data. The operation of transformation should preferably be simple as well as general enough in order to be able to reach the optimal triangulation from any initial one. The most natural transformation is the geometrical flip, or simply, the edge flip. The edge flip algorithm (EFA) was proposed by Charles Lawson in 1972 [Law72] for triangulations in the plane. It has also been shown that for any two triangulations T_1 and T_2 of a given planar point set V, there is always a sequence of Lawson's operations transforming T_1 into T_2. However, in space, the EFA produces self-intersections [AHA02]. Nevertheless, it is still possible to use it by introducing the next requirements [AvD02]:

Theorem 8.1.1 *The extended diagonal flip operation preserves two main topological characteristics, orientability and the genius, of the given class of triangulations of the same data set.*

The extended diagonal flip (EDF) allows self-intersections in a triangulation and defines a local flipping procedure for a nonflat quadrilateral such that the EDF might even produce the "gluing" of adjacent triangles (for more detail see [Alb03, AvD02]).

Thus, in the remaining part of this thesis we will use diagonal flip with requirements of the theorem 8.1.1, which we call EFA. EFA generates sequence of different triangulations $T_{ii=1}$, and even in some steps it might generate a triangulation T' which is already constructed, *i.e.*, $T' \in T_{ii=1}$. Generating the sequence of all different triangulations is only possible by applying an optimal criterion in EFA. Finally, we construct an optimal triangulation in accordance with the chosen criterion, In words, let $F(e)$ and $F(e')$ be values of the optimality functional of a triangulation before and after flipping edge e, the optimality criterion is $min\{F\}$. Then the edge e is flipped to the edge e' if and only if $F(e) > F(e')$.

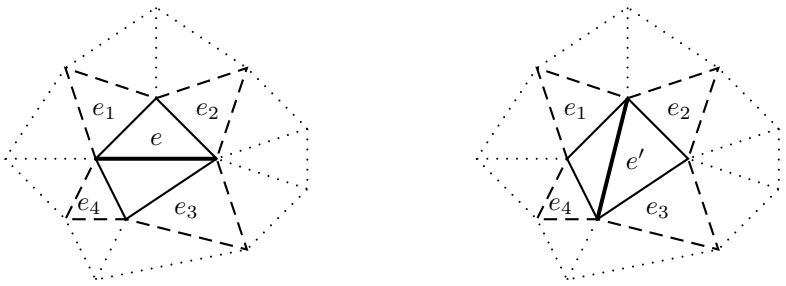


Figure 8.1: Edge flipping

The other idea is to use multi–edge flipping, *i.e.*, after flipping the edge e (it becomes e') and calculating $F(e')$ we continue by flipping the edges adjacent to e', *i.e.*, e_1, e_2, e_3 and e_4 (see figure 8.1) and calculate $F(e_i)$. Then, we choose the best triangulation from $F(e)$, $F(e')$ and $F(e_i)$ $(i = 1..4)$ in accordance with the criterion. This procedure we call multi–edge flipping algorithm (MEFA). Let us explain this algorithm in detail and why it should be used.

Suppose there is a triangulation $T(e)$ which has as value κ $(\kappa = F(e))$ for the chosen optimality criterion and we want to swap an edge e for choosing a better one in the sense of this optimality criterion. After flipping, let e become e' and the new triangulation $T(e')$ has the value κ'. If $\kappa > \kappa'$ then the edge e must be swapped in accordance with the optimality criterion (otherwise e is not swapped).

When EFA is applied, the next step considered would be flipping the edges e_i $(i = 1..4)$ adjacent to e' (it was e before flipping) and the result will be the same as by applying MEFA with two consecutive swapped edges e and e_i. However if $\kappa < \kappa'$ then the triangulation $T(e')$ with swapped edge e' is not considered at all. In MEFA we continue to swap the edges adjacent to e' of the triangulation $T(e')$. Then, by flipping the edge e_i and by consecutive flipping the edge e_j of $T(e')$ we construct new triangulations $T(e_i)$ and $T(e_i, e_j)$ with values κ_i and $\kappa_{i,j}$ respectively $(i \neq j,\ i, j = 1..4)$ and find a triangulation T''

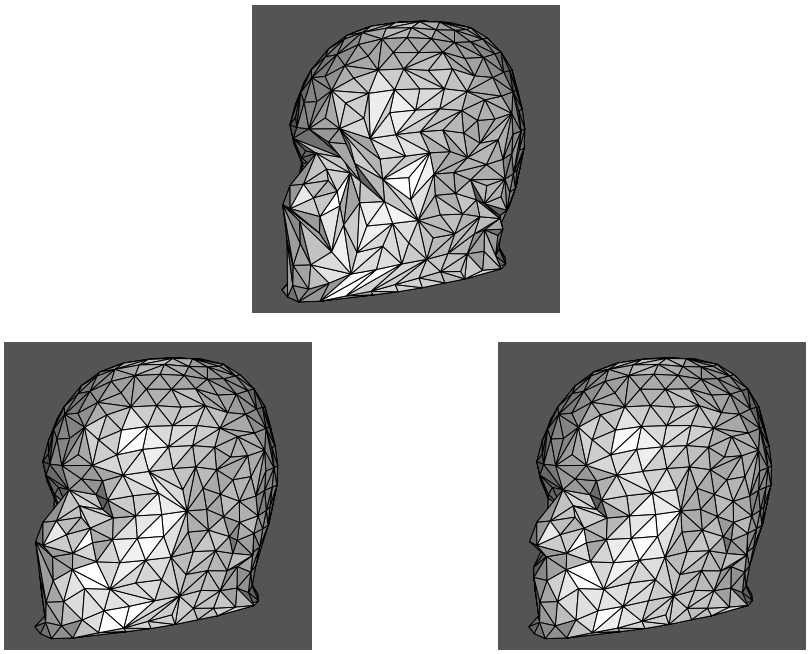


Figure 8.2: Bad initial triangulation (above) and two triangulations (below) after applying the mean criterion with EFA (left) and MEFA (right), resp.

with $\kappa'' = min_{(i\neq j,i,j=1..4)}(\kappa_i, \kappa_{ij})$. Thus, the scheme of MEFA is

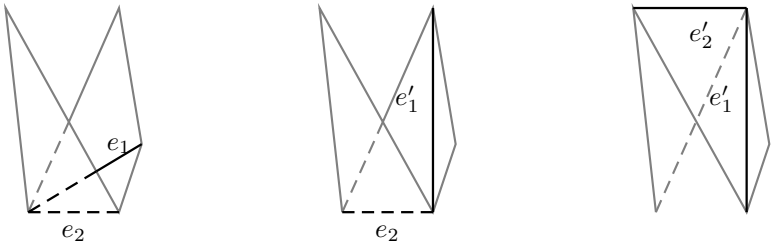
$$T(e) \rightarrow T(e') \rightarrow T(e_i) \rightarrow T(e_i, e_j).$$

Finally, to find an optimal triangulation we have to consider the next three possibilities:

1. Let be $\kappa < \kappa'$ and $\kappa < \kappa''$.
2. Let be $\kappa' < \kappa$ and $\kappa' < \kappa''$.
3. Let be $\kappa'' < \kappa$ and $\kappa'' < \kappa'$.

In the first case triangulation $T(e)$ is locally optimal and generation of $T(e')$ by EFA and T'' by MEFA are not needed in the construction. As a result, $T(e)$ will remain unchanged. In the second case triangulation $T(e')$ is locally optimal and it is generated by EFA as well as by MEFA. In the third case triangulation T'' is locally optimal and it is constructed by both EFA and MEFA if and only if $\kappa' < \kappa$. Otherwise, if $\kappa < \kappa'$, T'' is constructed only by MEFA.
As a result, MEFA for any optimality criterion usually gives better results than EFA, because multi–edge flipping produces triangulations which can not be

generated by using EFA with the same optimality criterion. For example, we consider figure 8.2, where the initial triangulation is the left triangulation in figure 7.6. Applying the optimality criterion with EFA (see figure 8.2 (left)) and MEFA (see figure 8.2 (right)) independently to this initial triangulation (in the top of figure 8.2) gives a "good–looking" optimal triangulation only by applying MEFA.



(a) (b) (c)

Figure 8.3: MEFA

Our experimental results show that MEFA is better than EFA and by using that approach it is possible to generate a "good" optimal triangulation from a given "not good" initial one. As an example let us consider a part of triangulation with gray edges in figure 8.3(a). After applying EFA, the edge e_1 flips to the edge e_1', the new triangulation (see figure 8.3(b)) is not optimal, therefore the edge e_1 must not be flipped. After applying MEFA the edge e_1 flips to e_1' and then the edge e_2 flips to e_2'. The new triangulation (see figure 8.3(c)) is optimal. Loosely speaking, MEFA is able to move from one local minimum to another one, which is better, in the case if the second local minimum is closed enough to the first one. Unfortunately, MEFA performs flipping a number of edges per turn[1],but in a certain cases it is not not necessary. The other problem we formulate as an open problem of this chapter:

Open Problem 8.1.2 *How many edges in MEFA are necessary to swap for avoiding useless calculation and generating a "good" initial triangulation?*

Remark 8.1.3 *The answer to that problem can not be general for any initial triangulation, because it depends on the "badness" of a triangulation. Therefore, we can reformulate the problem as follows: "Given an initial triangulation. How many edges has to be swapped per turn for avoiding unnecessary calculation and generating a "good" initial triangulation?"*

[1]We consequently flip maximally three edges in each step.

8.2 Constructing an initial triangulation

8.2.1 First set up of an "initial triangulation"

A very difficult problem in surface reconstruction is to generate a "good" initial triangulation for a given, but arbitrary, data set.
A simple method which can be used is to take first any four points, to construct a tetrahedron and then add recursively a new point one by one to one, two or more adjacent faces of the triangulation at once. The theoretical aspects of this recursive method were described in section 3.2.
In practice the number of faces of a triangulation to which a new point can be added is restricted by "visibility", as we are dealing with geometric properties of triangulations in this part of the thesis.

Definition 8.2.1 *Under visibility of a new point v with respect to a face $v_1v_2v_3$ of triangulation T one understands that the triangulation T and the tetrahedron $vv_1v_2v_3$ are intersecting only through face $v_1v_2v_3$ and there are no other intersection points.*

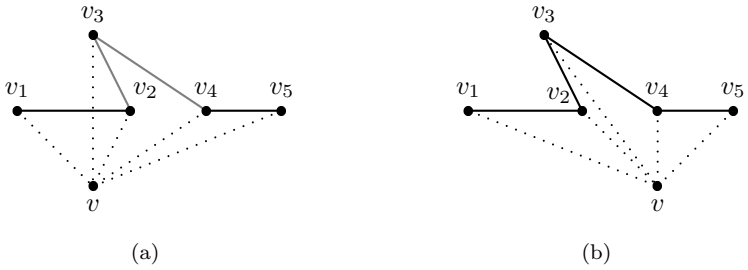


Figure 8.4: Black faces are visible and gray faces are invisible for a vertex v

An example of "visibility" (in **2D**) is clarified in figure 8.4. As one can see in figure 8.4(a) there are two visible (black) and two invisible (gray) faces whereas in figure 8.4(b) all faces are visible for v. In figure 8.4(a) face v_3v_4 is invisible because triangle vv_3v_4 is intersected by face v_1v_2. For the construction of a triangulation by recursively adding a new point one needs to find all visible faces of a triangulation at every step of the recursion.
Suppose at a certain step we generate a triangulation T_k of k vertices and we want to add a new point v for constructing T_{k+1}. Let all faces f_i $(i = 1..l)$ of T_k, that are visible for v, be found. The next question is to which face or faces a new point has to be added for generating a "good" triangulation. The simplest way is to add to a randomly chosen visible face at each step of the recursion. A typical result of a final triangulation is the tree–like triangulation shown in figure 8.5. We might try to improve our strategy in the following way.

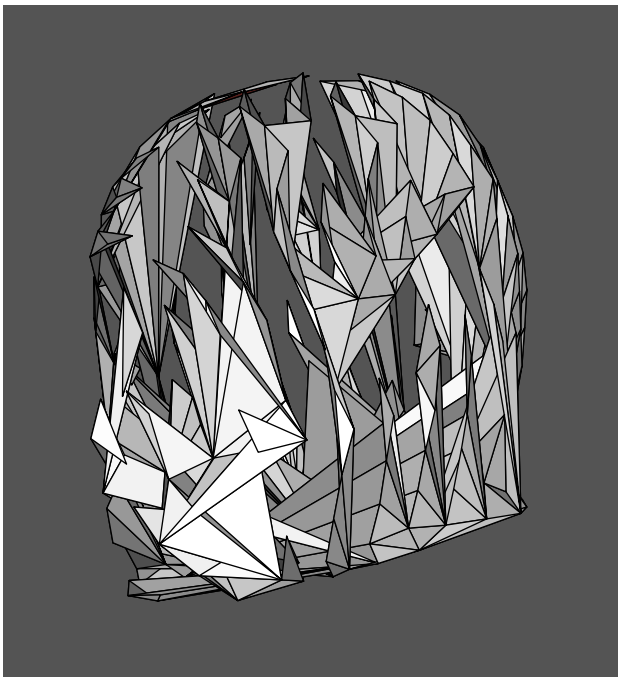


Figure 8.5: A tree-like triangulation in **3D**

We add a new point to faces which form a closed simple polygon by deleting their common edges (see section 3.2).

Definition 8.2.2 *A visible empty cycle is a closed simple polygon which is formed by visible faces.*

Then the next two problems appear. First, to define all visible empty cycles (the number of them is large) and second, which visible empty cycle has to be chosen for adding a new point[2].
Because finding empty cycles is a complicated work we prefer to construct a triangulation by adding a new point only to one visible face (flipping its edges or the edges adjacent to it can be considered as adding a new point of degree four, five and etc. to triangulation). In this case adding a point depends on the position of a new vertex with respect to triangulation T_k. There are four possibilities:

1. A new point coincides with a vertex of the triangulation;

[2]The number of empty cycles for a triangulation can be more than one. For example, two visible faces which are not connected by any "visible walk", where nodes of a "visible walk" are considered as visible faces.

2. A new point belongs to an edge of the triangulation;

3. A new point belongs to a face of the triangulation;

4. A new point does not belong to the triangulation.

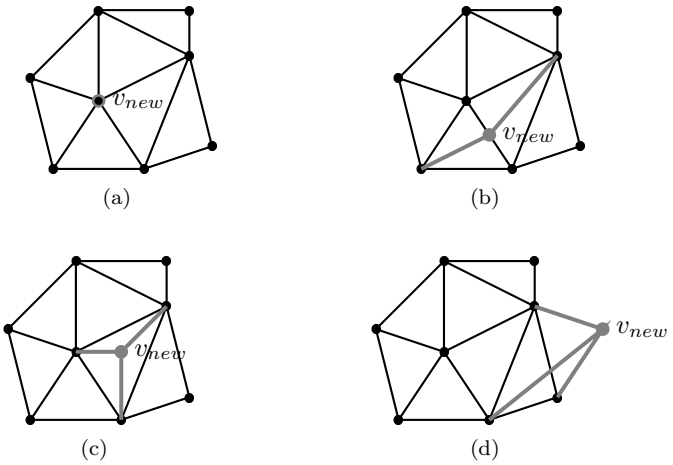


Figure 8.6: Position of a new point v_{new} with respect to the triangulation

In the first case we do not consider a new point anymore (see figure 8.6(a)). In the second case a new point has to be added to two triangles with a common edge which contains the new vertex (see figure 8.6(b)). In the third case a new vertex has to be connected to three vertices of the face (triangle) to which it belongs (see figure 8.6(c)). The last case is the only non trivial one (see figure 8.6(d)). A new vertex will be added to a visible face of a triangulation. It seems sufficient to find any visible face. Then, in order to avoid generation of a tree–like triangulation we apply EFA or MEFA with some optimality criterion after every step of the recursion. As a result, an initial triangulation is constructed recursively by adding a new point of degree three to a visible face and applying EFA or MEFA with some optimality criterion in every step. The general construction scheme is:

Algorithm 8.2.3 *Constructing an initial triangulation*

Step 1.
Given the data set $V = \{v_i\}_{i=0}^{N}$, *construct the tetrahedron* $T_4 = v_1v_2v_3v_4$, $k = 4$;

Step 2.
$k = k + 1$. *Construct* T_k *by adding* v_k *to a visible face* f *of* T_{k-1};

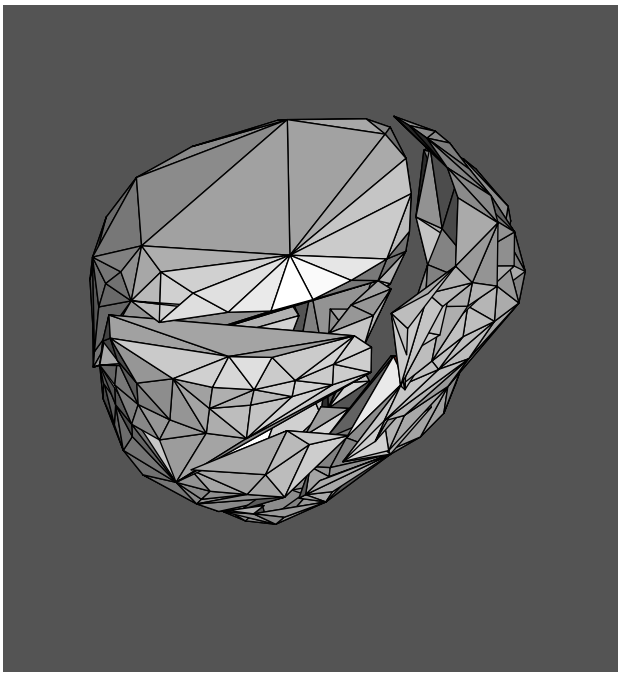


Figure 8.7: Triangulation is constructed directly from given data set

Step 3.
Apply the optimality criterion to T_k; if $k < N$ goto step 2;

This way of generating an initial triangulation is good in theory but not in practice, because the generated triangulations depend on the order in which points are taken from the given data set. Let us given a data set of an object for which we want to construct an initial triangulation. Let all data of set V be taken from k parallel slices which holds, for example, in the case of laser scanning of an object. Then, the data set is the union of the data subset of every slice

$$V = \cup_{i=1}^{k} V_i,$$

where $V_i = \{v_j^i\}_{j=1}^{l_i}$ and l_i is the number of points in i-th slice. If we take such a data set for generating an initial triangulation, then by the algorithm 8.2.3 we take the first four points $v_j^1 \in V_1$ $(j = 1..4)$ for constructing an initial tetrahedron. But these points are possibly in the same plane; therefore in the beginning we have to choose at least four points which are not in one plane. Let they be v_1^1, v_2^1, v_3^1 and v_j^i $(i = 2..k)$. Suppose four points are found and the tetrahedron T_4 is constructed. In the next step we add a new point $v^* \in V \backslash v(T_4)$ $(v(T_4) = \{v_1^1, v_2^1, v_3^1, v_j^i\})$ to a visible face of T_4 and produce T_5. Then we apply

EFA or MEFA with respect to the chosen optimality criterion. Following this way, one constructs triangulations

$$T_4 \to T_5 \to T_6 \to \cdots \to T_N.$$

An example of using this way of generating a triangulation for the data set, which is taken from parallel slices of laser scanning of a brain, is shown in figure 8.7. As can be seen this is not a proper triangulation. This happens because a number of added vertices belong to the same slice, where vertices are consequently close to each other. This gives a triangulation which look like a "snake" and the final one is considered as a "good" initial triangulation. Such a triangulation can be transferred to a "good–looking" triangulation, but for that we have to use MEFA with a large number of simultaneously flipping edges. In that case the number of operation will be too large. As a result, we need a different algorithm for generating an initial triangulation or to reorder the data set before applying algorithm 8.2.3.

8.2.2 The effect of reordering data

Let us, before generating a triangulation, reorder the points $V = \{v_i\}_{i=0}^{N}$ of the data by the following algorithm:

Algorithm 8.2.4 *Reordering the data set*

1. *Find a vertex v^* which is maximally outlying with respect to all other vertices:*
$$v^* = \{v \in V | \max_{i=0..N} d(v, v_i)\},$$
where $d(v_i, v_j)$ is the distance between two vertices v_i and v_j;

2. *Put the other points in decreasing order by distance between v^* and $v_i \in V \setminus v^*$. Then the reordered data set will be*
$$V' = \{v_1, \ldots, v_j, v_{j+1}, \ldots\},$$
where
$$d(v^*, v_1) \geq \cdots \geq d(v^*, v_j) \geq d(v^*, v_{j+1}) \geq \ldots$$

After such reordering of the data set we start to construct a triangulation by algorithm 8.2.3. We take the first three points $v_j \in V'$ $(j = 1, 2, 3)$ and construct a tetrahedron $T_4 = v^* v_1 v_2 v_3$. Then we take the next point v_4 of the reordered data set and find for it a close visible face of T_4. Adding v_4 to the face gives T_5. After applying the mean curvature criterion, T_5 becomes T_5'. Continuing in the same way we construct the final triangulation T_N', which is optimal with respect to the mean curvature criterion. The algorithm for this construction is as follows:

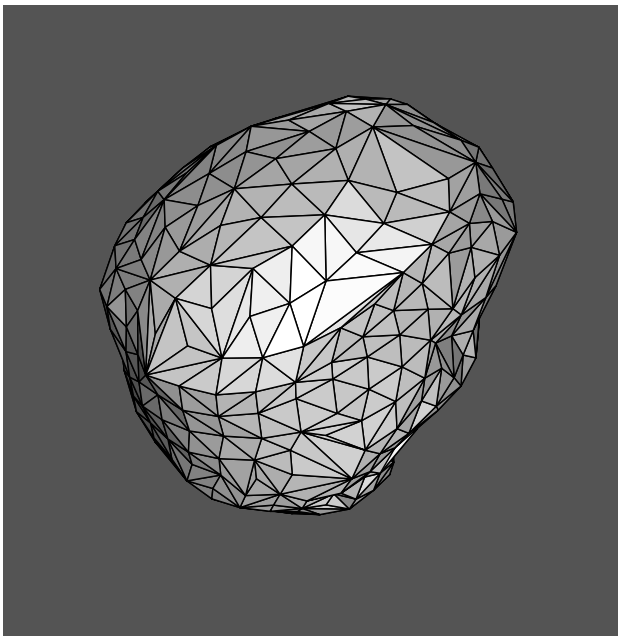


Figure 8.8: Triangulation, constructed from a reordered data set

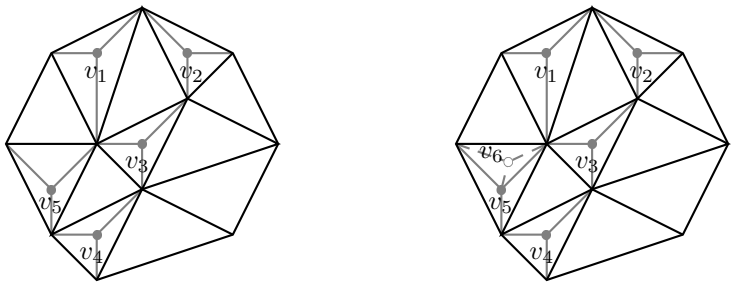
Algorithm 8.2.5 *Constructing an initial triangulation*

1. *Given a data set* $V = \{v_i\}_{i=0}^{N}$. *Reorder it by algorithm 8.2.4;*
2. *Construct the tetrahedron* $T_4 = v^* v_1 v_2 v_3$, $k = 4$*;*
3. $k = k + 1$*;*
4. *Find a visible face* f *of* T_{k-1} *and construct* T_k *by adding* v_k *to* f*;*
5. *Apply the optimality criterion to* T_k*;*
6. *If* $k < N$ *goto step 3;*

The effect of reordering the data set before generating a triangulation can be seen in figure 8.8, where we take the same data set of the brain as in the previous subsection. If we compare the triangulations of figure 8.7 and figure 8.8 you can see that the reordering effect gives certainly a better result than was obtained in the previous subsection.

8.2.3 The effect of adding a number of points per time

In the previous subsection the criterion of optimization is applied in every step of the process, that is not efficient because the main part of a triangulation remains



(a) (b)

Figure 8.9: Adding a number of new points to a triangulation per time

unchanged. Therefore we decided to apply the criterion only after adding several new points, *i.e.,* to a triangulation T_k we add l new points to visible faces, and after that an optimality criterion will be applied. Every new point v_i $(i = 1..l)$ should be added in such a way that a visible face for v_i does not contain the vertices $v_1, .., v_{i-1}$. Otherwise, the resulting triangulation can partially look like a tree (see figure 8.7). An example of adding l new points to a triangulation is given in figure 8.9. In figure 8.9(a) the gray dotted points $v_1, .., v_5$ indicate the points which are correctly added to a triangulation. In figure 8.9(b) the white point v_6 (with dashed edges) is added to a face of a triangle which is constructed by adding a point at the same step, and hence forbidden. The other condition required for adding a new point is that every new point is added to a visible face of the vertex v_* of algorithm 8.2.4. This way of adding new points we called *constructing a "carcase"*, where under a "carcase" we understand the object which is constructed by adding a number of points per time to visible faces of the fixed vertex v_*. It can be a final triangulation if all vertices are added, or it can be a halfway triangulation if only a part of all vertices are added. As a result, we first construct a carcase and then, if it is a halfway triangulation, we add the remaining new points to it. Therefore, the generation algorithm can be divided into two steps:

Algorithm 8.2.6 *Constructing an initial triangulation by adding a number of points per time*

1. **Step 1.** *Constructing a carcase (see example in figure 8.10).*
   (a) *Given a data set of vertices* $V = \{v_i\}_{i=0}^{N}$. *Reorder them by the above described rules.*
   (b) *Construction of the tetrahedron* $T_4 = v^* v_1 v_2 v_3$, $k = 4$.
   (c) *Adding* l *new vertices* v_l *to the visible faces of* T_k *(visible faces must not contain a vertex* v_l*) and constructing* T_{k+l} *(not all new vertices can be added; those which are not added we keep in a data set* W*).*

(d) *Applying the optimality criterion to* T_{k+l}, $k := k + l$, *if* $k < N$ *goto step 1c;*

2. **Step 2.** *Adding the remaining points (see the example in figure 8.11).*

   (a) **If** $W \neq 0$ **then** $V = W$ *and add* l *new vertices to a visible face of* T_k *(here there is no restriction w.r.t. the visible face),* **else** *stop algorithm.*

   (b) *Applying the optimality criterion to* T_{k+l}, $k := k + l$, *if* $k < N$ *goto step 2a;*

   (c) *An initial triangulation* T_N *of* N *vertices is constructed.*

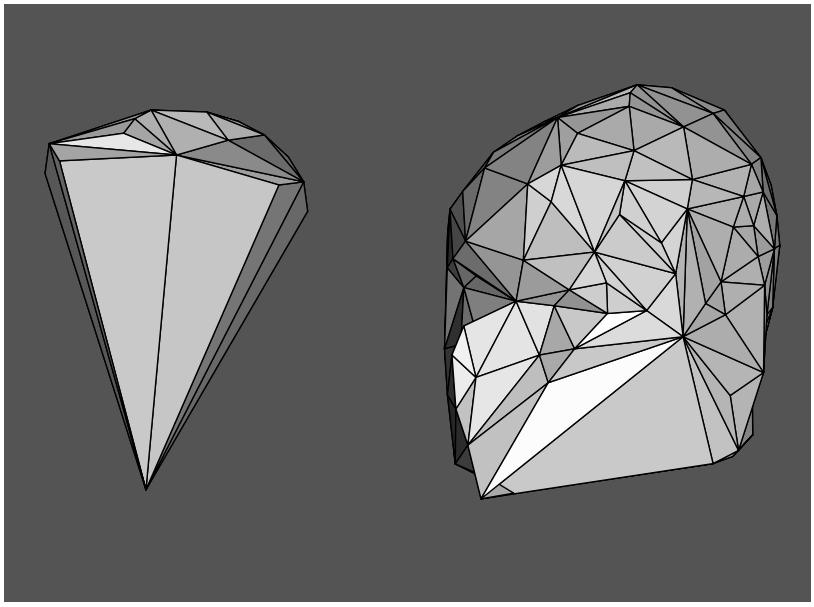


Figure 8.10: Constructing a carcase by adding a number of points and applying the mean curvature criterion

Above it was mentioned that new points are added to visible faces. Below we want to specify the way of adding new points but before that we first introduce four types of "visibility". Under type of "visibility" we understand that the face is visible by definition 8.2.1, and some additional property for a new point and a visible face holds. We define four types of "visibility":

1. **"Visibility" is "perpendicular to face"**

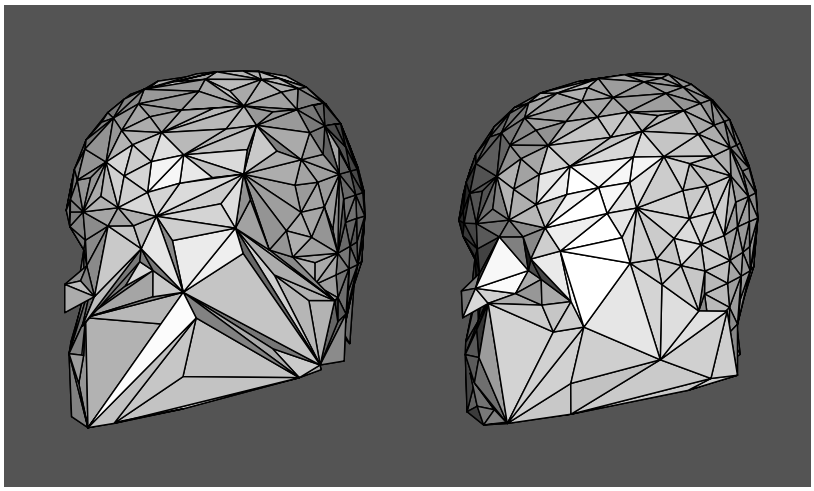


Figure 8.11: Adding the remaining points to the carcase (left) and applying the mean curvature criterion (right)

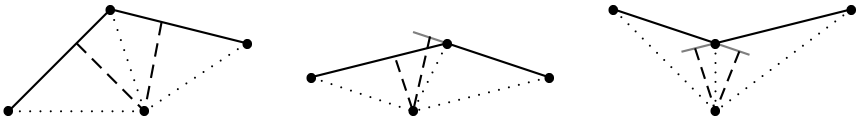


Figure 8.12: "Visibility" as perpendicular to the face. Dashed lines are perpendicular to faces, dotted lines are supposed to be connecting edges to a new point

The perpendicular line, going from a new point to a visible face, intersect the face, *i.e.*, the intersection point is inside the face. There are three possibilities:

(a) It holds for two adjacent faces as in figure 8.12(a), then a new vertex is added to these two faces with deleting their common edge.

(b) It holds only for one face as in figure 8.12(b), then we take it;

(c) Both faces are visible by definition 8.2.1 but the perpendicular lines to the faces do not intersect them as in figure 8.12(c), then a new vertex is added to these two faces with deleting their common edge.

2. **"Visibility" is "sharp angles of three new triangles"**

All three new faces, which are constructed by adding a new point to a visible face, are acute triangles. Examples are in figure 8.13, where in figure 8.13(a) a new point can be added to a face, whereas in figure 8.13(b) it can not.

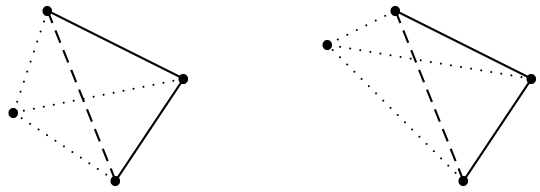


Figure 8.13: "Visibility" as sharp angles of three new triangles

3. **"Visibility" is "minimum distance to vertices of triangle"**
   The sum of the distances between three vertices of a visible face (triangle) and a new point is minimal.

4. **"Visibility" is a regular visible face**
   A face is visible by definition 8.2.1, *i.e.*, there is no additional condition.

We describe the way of adding a new point to a visible face for algorithm 8.2.6:

> **For Step 1.** At first we find the closest vertex of the star of v^* to a new point v_{new}. Suppose it is vertex v'. If the new edge $v'v_{new}$ does not intersect the triangulation, then we define which faces f_1 and (or) f_2 are visible from v_{new}, where f_1 and f_2 are triangles with common edge v^*v'. Then a new point is added to any visible[3] face.
>
> **For Step 2.** Adding the remaining points to a carcase (the triangulation is constructed by *step 1*) proceeds by the next rules. At first the closest point should be found - suppose it is v'. If the new edge $v'v_{new}$ does not intersect the triangulation, then we deduce which faces f_i of star v' are visible[4] from v_{new} and choose only one.

The method described above constructs a "good" initial triangulation for the given data set of an object which is homeomorphic to the **2D** sphere. It contains three steps:

1. reordering the data set of points;

2. constructing a carcase;

3. adding the remaining points to the carcase.

[3] In this case "visibility" means "perpendicular to face".

[4] In this case "visibility" means at first "perpendicular to face", at second - "sharp angles of three new triangles", at third - "minimum distance to vertices of triangle" and at fourth - "if it is possible".

In section 8.3 examples of the step by step construction of an initial triangulation is given. In figure 8.14- 8.19 initial triangulations are constructed by adding all possible new vertices in constructing the carcase. In figure 8.20- 8.25 initial triangulations are constructed by adding a few new vertices in constructing the carcase.

8.3 Conclusion

In this chapter we have shown which advantages and disadvantages MEFA has over EFA. We have also described algorithms for generating an initial triangulation for a given data set. The algorithm is not perfect yet, because for some examples a "cleft" region appears on the constructed surface, for example for the data set of the dumb-bell 8.25. The method constructs a "good" initial triangulation for most of the surface, but near the small ball there is a "cleft". A "cleft" region also appears for the data set of the scalp (see figure 8.16 in the right). The "cleft" can be corrected by applying MEFA with a large number of flipping edges. In such a case the number of flipping edges for making the proper correction of the "cleft" region is roughly defined by the number of edges in that region. Another specific property of the method is that the appearance of a "cleft" region depends on the number of points which are added per time in constructing the carcase. As an example we consider the resulting triangulations of the same data set which are constructed by using a different number of added points per time (see figure 8.16 and 8.22). Finally we formulate two open problems of this chapter:

Open Problem 8.3.1 *Let an initial triangulation have a "cleft" region. To find an algorithm for correcting the "cleft" of a triangulation such that it results in a "good" initial one.*

Open Problem 8.3.2 *To study the dependence between the number of added points per time and the appearance of "cleft" regions in the method. To find the optimal number of points to be added per time for constructing a "good" initial triangulation.*

Appendix

In this section we consider step by step the algorithm 8.2.6[5] for constructing an initial triangulation on a real data set of a scalp and a dump-bell. Before starting the addition of new vertices, the data set should be reordered and an initial tetrahedron should be constructed.

[5] In the appendix algorithm refers to algorithm 8.2.6.

First, we describe the construction of an initial triangulation by adding all possible points to it per turn in *step 1* of the algorithm. All the remaining points of the data set are added to a visible face of the triangulation one by one. As a result, a triangulation for the data set of a scalp (dump-bell) is shown at the top of figure 8.14 (see figure 8.17). After applying the mean curvature criterion the triangulation becomes as given at the bottom of figure 8.14 (see figure 8.17). Now, a carcase for the data set of the scalp (dump-bell) in *step 1* of algorithm 8.2.6 is constructed and *step 2* should be run. Adding new points gives a triangulation as given at the top of figure 8.15 (see figure 8.18) and after applying the mean curvature criterion it becomes as given at the bottom of figure 8.15 (see figure 8.18). Continuing *step 2* of the algorithm gives the triangulation of the data set of the scalp (dump-bell) in figure 8.16 (see figure 8.19).
Now we consider the construction of an initial triangulation by adding ten points to it per turn in *step 1* of the algorithm, and then applying the mean curvature criterion. As a result of *step 1*, the algorithm constructs a carcase for the data set of the scalp (dump-bell) which is given in figure 8.20 (see figure 8.23). Then the algorithm runs *step 2*, where all possible remaining points are added to the triangulation. The result of the first turn of the algorithm for the data set is presented in figure 8.21 (see figure 8.24) and the resulting triangulation is shown in figure 8.22 (see figure 8.25).
As a result, using in *step 1* of the algorithm the condition, that all possible points should be added to a triangulation, a "good" triangulation is obtained without cleft region for the dump-bell (see figure 8.19) and a "bad" triangulation with a cleft region for the scalp (see figure 8.16), whereas the condition that only ten points are added in every turn of *step 1* gives a "bad" triangulation for the dump-bell (see figure 8.25) and a "good" triangulation for the scalp (see figure 8.22). Therefore in section 8.3 we formulated two open problems of this chapter.

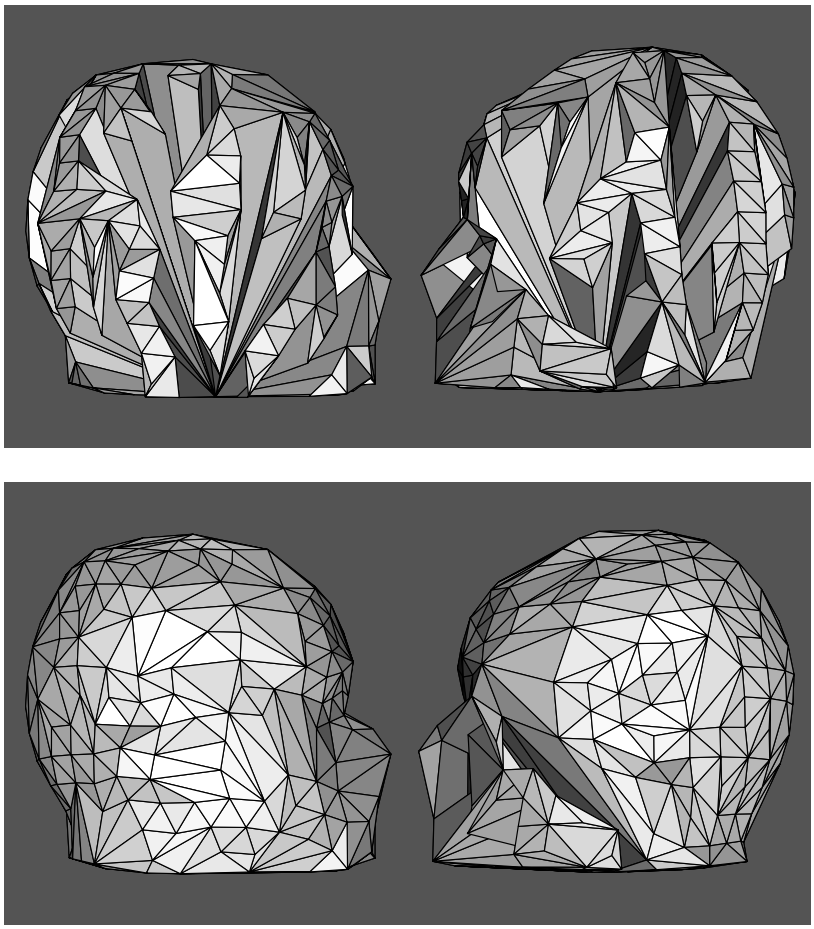


Figure 8.14: A carcase is constructed by adding all possible points (top) and applying the mean curvature criterion (bottom)

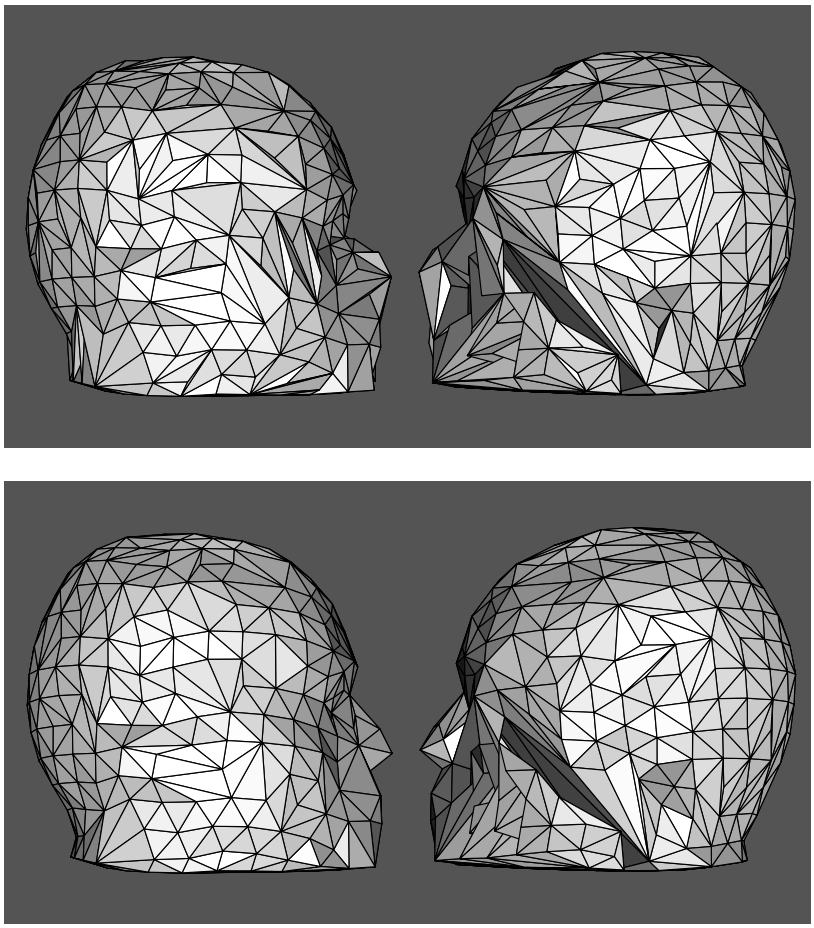


Figure 8.15: Adding the remaining new points to the carcase (top) and applying the mean curvature criterion (bottom)

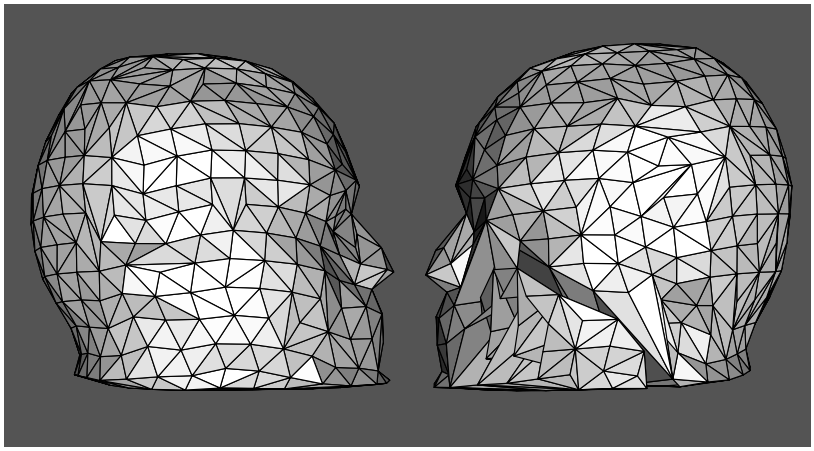


Figure 8.16: Resulting triangulation

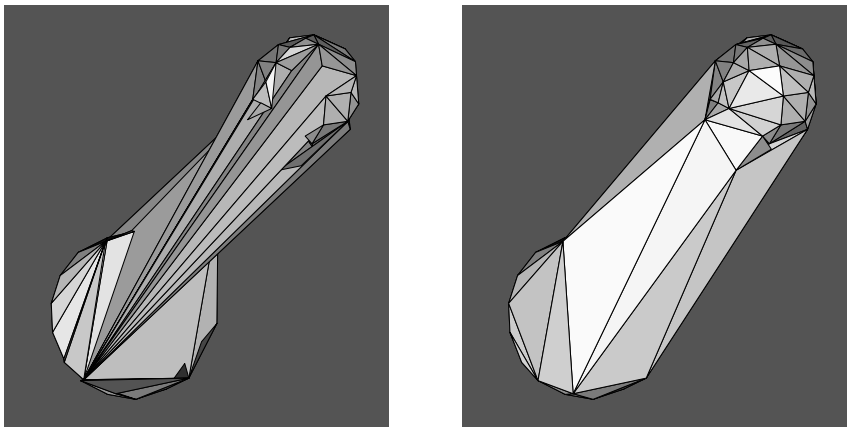


Figure 8.17: A carcase is constructed by adding all possible points (left) and applying the mean curvature criterion (right)

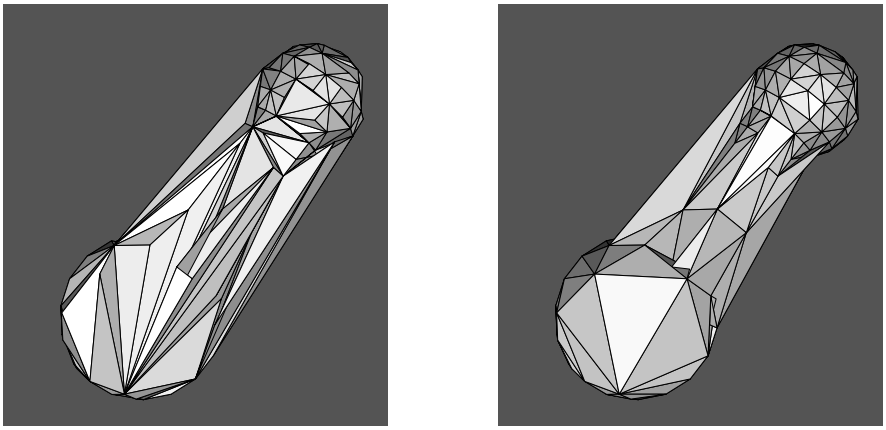


Figure 8.18: Adding the remaining new points to the carcase (left) and applying the mean curvature criterion (right)

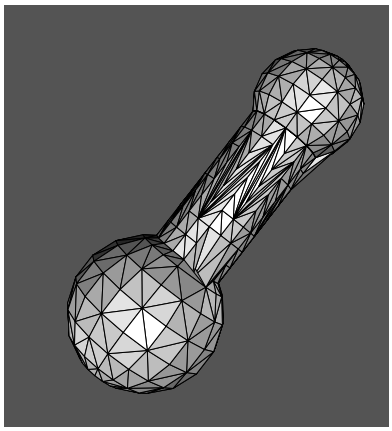


Figure 8.19: Resulting triangulation

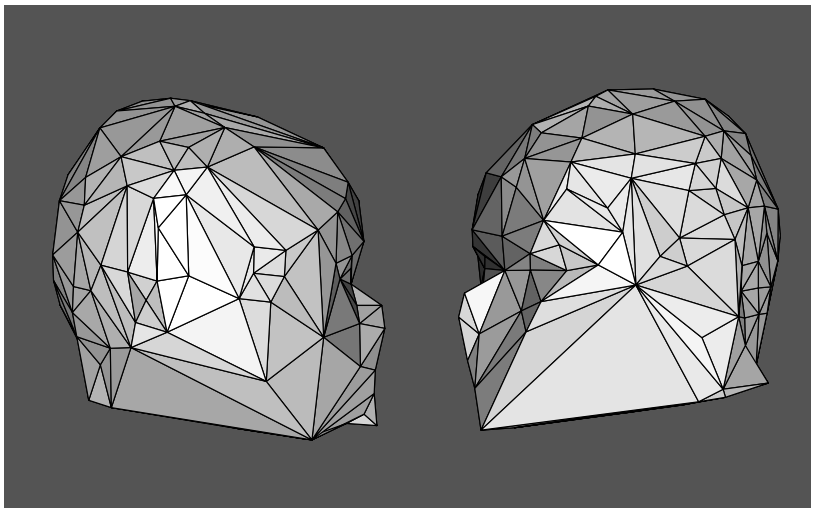


Figure 8.20: A carcase is constructed by adding a few possible points and applying the mean curvature criterion

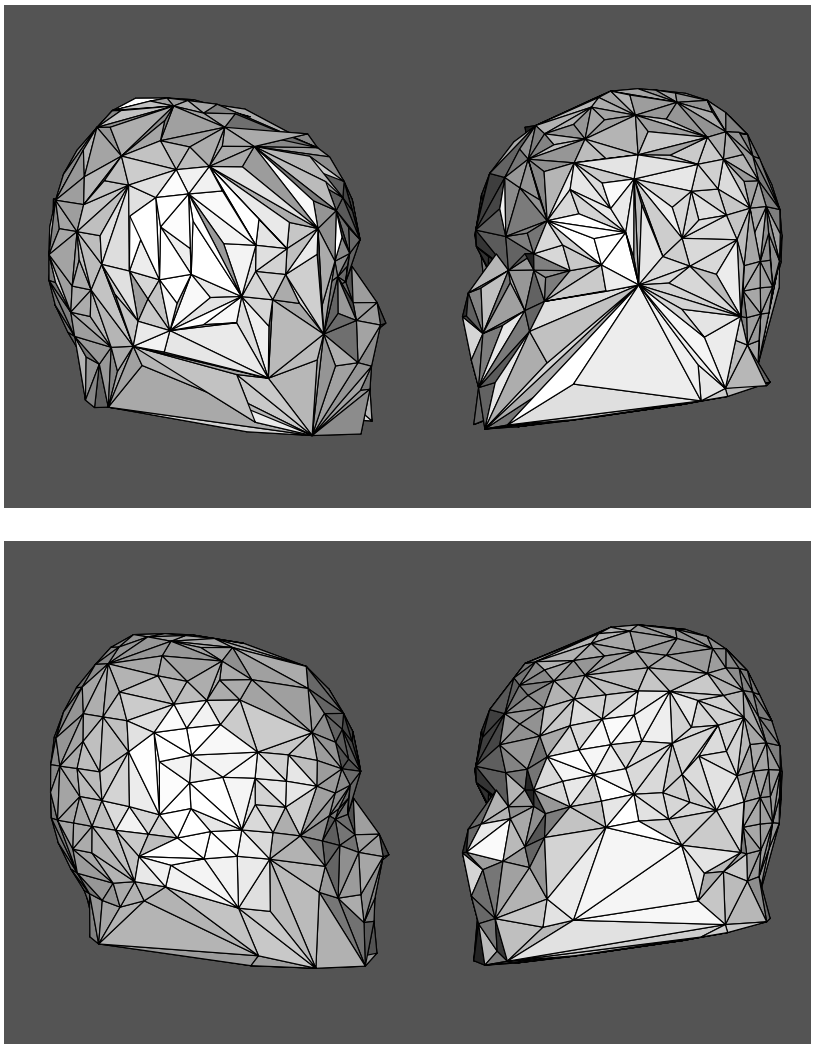


Figure 8.21: Adding the remaining new points to the carcase (top) and applying the mean curvature criterion (bottom)

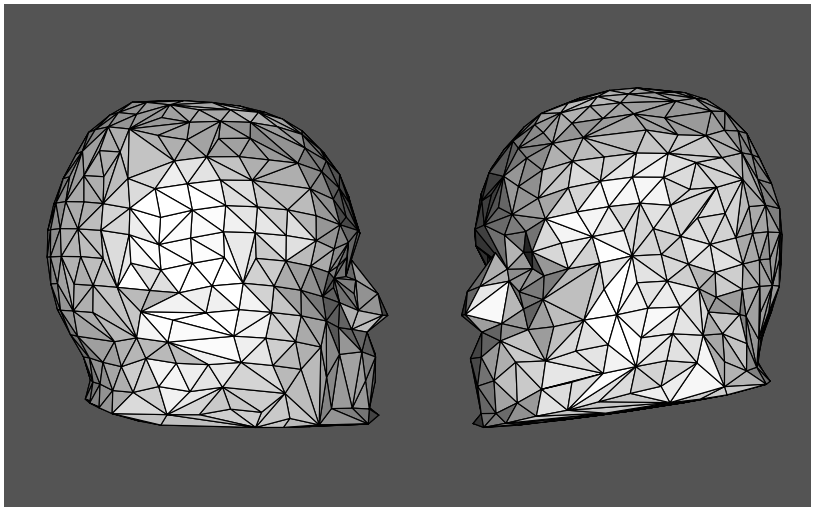


Figure 8.22: Resulting triangulation

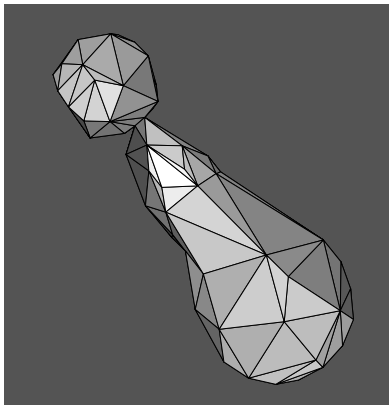


Figure 8.23: A carcase is constructed by adding a few possible points and applying the mean curvature criterion

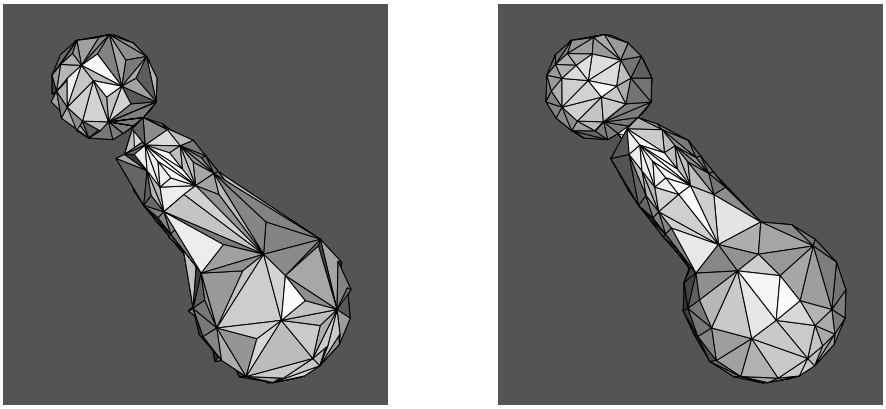


Figure 8.24: Adding the remaining new points to the carcase (left) and applying the mean curvature criterion (right)

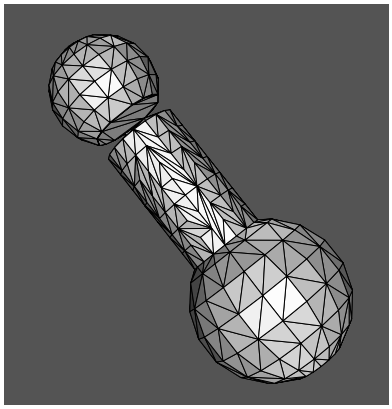


Figure 8.25: Resulting triangulation

Chapter 9

Future work

9.1 Main notions

In part II of the thesis we described a method for generating an initial triangulation and we showed the various steps of the construction in a number of examples. Because the method does not work well in some cases, below we describe our ideas for constructing a good initial triangulation for any kind of object homeomorphic to the sphere. As we said in the previous section, "cleft" regions can appear in the construction. Therefore we want to find a method for defining such regions and correcting them. Another interesting problem is to find the dependence between the number of added points per step (in constructing the carcase) and the quality of the resulting initial triangulation. Defining that dependence is an interesting task because, for example, constructing a good initial triangulation of the scalp and the dump-bell data set, while adding a few new points per step, gives a good triangulation for the scalp but results in a triangulation with "cleft"–region for the dump-bell, whereas adding all possible new points in one step gives a triangulation with "cleft"–region for the scalp and a good triangulation for the dump-bell! What is a very strange phenomenon.

Our ideas for solving the problem of "cleft" regions are as follows. First suppose that a point to be added should be in the neighborhood of other points. One way is to find the mean distance between all existing points of the data set and to take it as a radius R_{mid} of a sphere in which such a point must be lying. Then a new point should be added only to points which are in that neighborhood. Also the edges of the final triangulation should not be longer than R_{mid}. So flipping should be done in such a way that long edges are avoided. Second idea is to distinguish a "cleft" region and to correct it. Suppose there is a "cleft" region. Then we define points close to each other which are not connected and define an empty cycle within them. After that we can delete internal edges

and retriangulate the empty cycle such that close points are connected. A third idea, and simple one, is to construct the carcase by adding only points which are outside of the already constructed object. This idea is similar to the construction of the convex hull of points which are taken as carcase.

9.2 Problems of generation and optimization

The following troubles were met in finding an optimal triangulation by using the mean and Gaussian curvatures:

1. **Problem of the mean curvature**
   Let us consider the triangulation in figure 9.1. This triangulation has an edge e_i which will be swapped to the edge e'_i by the flip algorithm: suppose $H(e_i)$ and $H(e'_i)$ are mean curvatures and $\beta(e_i)$ and $\beta(e'_i)$ are dihedral angles of the edges e_i and e'_i, respectively. After flipping, e_i becomes e'_i. If $||e_i|| \gg ||e'_i||$ then $H(e_i) > H(e'_i)$ even if $\beta(e_i) < \beta(e'_i)$. As a result there will be a self–intersection. It means that in the flipping algorithm an extra check for self–intersection should be done. The problem is what kind of check should be used in that case.

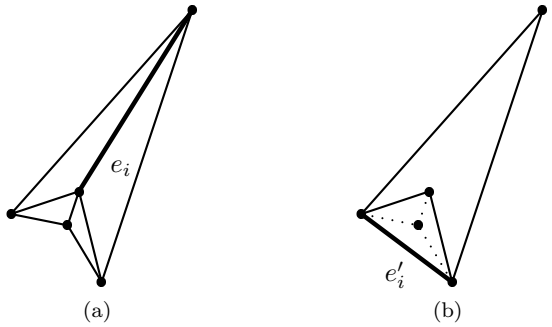


Figure 9.1: Before (left) and after flipping (right)

2. **Problem of Gaussian Curvature**
   Flipping an edge e_i gives a better triangulation, but a self–intersection will frequently appear (see figure 9.2). The problem is what kind of check should be used in that case. This problem was partially solved in the work of L. Alboul [AvD02, Alb03] for convex data, but is an open problem for non–convex data.

3. **Problem of flipping algorithm**
   Take a triangulation on which the optimality criterion reaches the value $F(T)$. Consider two edges e_1 and e_2, for which flipping gives the same

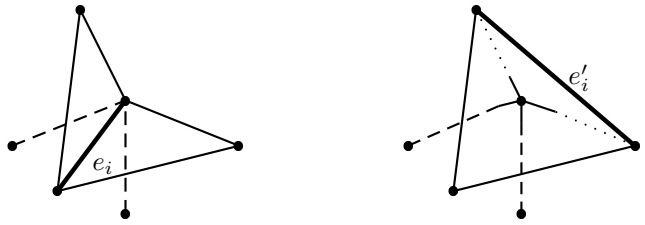


Figure 9.2: Before (left) and after flipping (right)

edge (see figure 9.3). Suppose after flipping the edge e_i, the triangulation has criterion value F_i $(i = 1, 2)$ and $F > F_i$. Then by EFA these two edges can be flipped, which gives an optimal triangulation, but flipping the edges e_1 and e_2 gives two different triangulations. In EFA the order of edges is arbitrary. For example, flipping e_1 could result in the global optimum, whereas flipping e_2 results in a situation that is locally optimal: hence the best triangulation is not found in such a case.

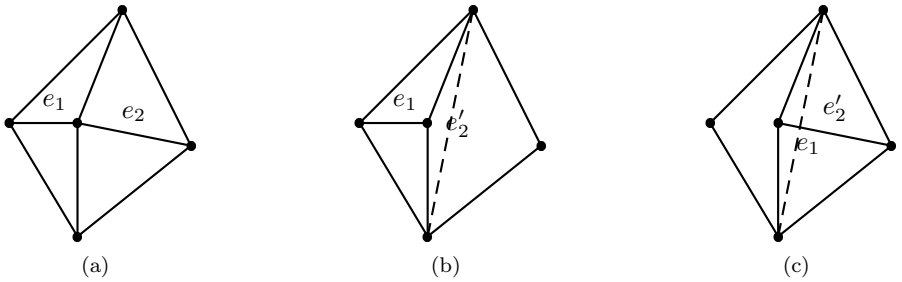


Figure 9.3: Before flipping (left), after flipping the edge e_2 (middle) and the edge e_1 (right)

Bibliography

[AB99] N. Amenta and M. Bern. Surface reconstruction by voronoi filtering. *Discrete and Computational Geometry*, 22(4):481–504, December 1999.

[AB02] D. Attali and J.-D. Boissonnat. A linear bound on the complexity of the delaunay triangulation of points on polyhedral surfaces. In *Proc. of the 7th ACM Symposium on Solid Modeling and Applications*, pages 139–145, Saarbrücken, Germany, June 2002.

[ABE98] N. Amenta, M. Bern, and D. Eppstein. The crust and the β-skeleton: combinatorial curve reconstruction. *Graphical Models and Image Processing*, 60/2(2):125–135, 1998.

[ABK98] N. Amenta, M. Bern, and M. Kamvysselis. A new voronoi-based surface reconstruction algorithm. In *Proc. of SIGGRAPH'98*, pages 415–421. ACM, 1998.

[AC99] N. Amenta and S. Choi. One–pass delaunay filtering for homeomorphic $3d$ surface reconstruction. Technical Report No. TR99-08, Univ. of Texas, Austin, Dept. of Computer Sciences, Taylor Hall 2.124, Austin, TX 78712-1188, USA, 1999.

[ACDL00] N. Amenta, S. Choi, T. K. Dey, and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. In *Proc. of the 16th ACM Symposium on Computational Geometry*, pages 213–222, June 2000.

[AdGB00] M. J. Abásolo, A. de Giusti, and J. Blat. A hierarchical triangulation for multiresolution terrain models. *The Journal of Computer Science and Technology*, October 2000. Special Issue on Computer Science Research: State of the Art.

[AER04] L. Alboul, G. Echeverria, and M. Rodrigues. Total absolute curvature as a tool for modelling curves and surfaces. In *Proc. of Grid*

Generation: Theory and Applications, page 12, Moscow, June 2004. Russian Academy of Sciences.

[AF92] D. Avis and K. Fukuda. Reverse search for enumeration. Research report No. 92-5, GSSM, Univercity of Tsukuba, April 1992.

[AFK99] N. Alt, U. Fuchs, and K. Kriegel. On the number of simple cycles in planar graphs. *Combinatorics, Probability & Computing*, 8(5):397–405, September 1999.

[AGJ02] U. Adamy, J. Giesen, and M. John. Surface reconstruction using umbrella filters. *Computational Geometry: Theory & Applications*, 21(1):63–86, January 2002.

[AHA02] O. Aichholzer, F. Hurtado, and L. Alboul. On flips in polyhedral surfaces. *International Journal of Foundations of Computer Science*, 13(2):303–311, 2002.

[AK96] D. Avis and C. M. Kong. Generating rooted triangulations with minimum degree four. In F. Fiala, E. Kranakis, and J.-R. Sack, editors, *Proc. of the 8th Canadian Conference on Computational Geometry*, Ottawa, August 1996. Carleton University.

[AKTvD99] L. Alboul, G. Kloosterman, C. R. Traas, and R. van Damme. Best data-dependent triangulations. Memorandum No. 1487, Faculty of Applied Mathematics, University of Twente, 1999.

[Alb03] L. Alboul. Optimising triangulated polyhedral surfaces with self–intersections. In N.J. Wilson and R.R. Martin, editors, *The Mathematics of Surfaces*, volume X, pages 48–73, Oxford, 2003. Clarendon Press. (LNCS 2768).

[Att98] D. Attali. r-regular shape reconstruction from unorganized points. *Computational Geometry Theory and Applications*, 10(4):239–247, July 1998.

[AvD95a] L. Alboul and R. van Damme. Polyhedral metrics in surface reconstruction: Tight triangulations. Memorandum No. 1275, Faculty of Applied Mathematics, University of Twente, 1995.

[AvD95b] L. Alboul and R. van Damme. Tight triangulations. In M. Dæhlen, T. Lyche, and L.L. Schumaker, editors, *Mathematical Methods for Curves and Surface*, pages 517–526, Nashville, 1995. Vanderbilt University Press.

[AvD96] L. Alboul and R. van Damme. Polyhedral metric in surface reconstruction. In G. Mullineux, editor, *The Mathematics of Surface*, volume VI, pages 171–200, Oxford, 1996. Clarendon Press.

[AvD97] L. Alboul and R. van Damme. Polyhedral metrics in surface reconstruction: Tight triangulations. In T. Goodman, editor, *The Mathematics of Surface*, volume VII, pages 309–336, Oxford, 1997. Clarendon Press.

[AvD02] L. Alboul and R. van Damme. On flips in polyhedral surfaces: a new development. In *Proc. of the 18th European Workshop on Computational Geometry*, pages 80–84, Warsaw, April 2002.

[Avi96] D. Avis. Generating rooted triangulations without repetitions. *Algorithmica*, 16(6):618–632, December 1996.

[AZ67] A. D. Aleksandrov and V. A. Zalgaller. *Intrinsic geometry of surfaces*. AMS, Rhode Island, 1967.

[Ban67] T. F. Banchoff. Critical points and curvature for embedded polyhedra. *Journal of Differential Geometry*, 1:257–268, 1967.

[BB97] F. Bernardini and Ch. L. Bajaj. Sampling and reconstructing manifolds using α–shapes. In *Proc. of the 9th Canadian Conference on Computational Geometry*, pages 193–198, 1997.

[BBX95] Ch. L. Bajaj, F. Bernardini, and G. Xu. Automatic reconstruction of surfaces and scalar fields from $3d$ scans. In *Computer Graphics Proceedings,Annual Conference Series*, Proc. of SIGGRAPH'95, pages 109–118. ACM SIGGRAPH, 1995.

[BE95] M. Bern and D. Eppstein. Mesh generation and optimal triangulation. In D.-Z. Du and F. K.-M. Hwang, editors, *Computing in Euclidean Geometry*, number 4 in Lecture Notes Series on Computing, pages 47–123. World Scientific, second edition, 1995.

[BF67] R. Bowen and S. Fisk. Generation of triangulations of the sphere. *Mathematics of Computation*, 21:250–252, 1967.

[BG92] J.-D. Boissonnat and B. Geiger. Three dimensional reconstruction of complex shapes based on the delaunay triangulation. Rapport de recherche de l'INRIA No. RR-1697, INRIA, Le Chesnay Cedex, May 1992.

[BK79] U. Brehm and W. Kühnel. Smooth approximation of polyhedral surfaces with respect to curvature measures. In *Global differential geometry*, pages 64–68, 1979.

[BK97] T. F. Banchoff and W. Kühnel. Tight submanifolds, smooth and polyhedral. volume 32, pages 51–118. MSRI Publications, 1997.

[BM76] J. A. Bondy and U. S. R. Murty. *Graph Theory with applications.* The Macmillan Press Ltd., London, 1976.

[BM01] G. Brinkmann and B. McKay. Plantri and fullgen, 2000–2001. http://cs.anu.edu.au/ bdm/plantri/

[Bob04] A. Bobenko. A conformal energy for simplicial surfaces. Technical report, Institut für Mathematik, Technishe Universität Berlin, Berlin, June 2004.

[Boi84] J.-D. Boissonnat. Geometric structures for three dimensional shape representation. *ACM Transactions on Graphics*, 3(4):266–286, 1984.

[Bou94] P. Bourke. Data reduction in terrain modelling, June 1994. http://astronomy.swin.edu.au/ pbourke/terrain/terraindata

[BP71] L. W. Beineke and R. E. Pippert. The number of labelled dissections of a k–ball. *Mathematische Annalen*, 191:87–98, 1971.

[BP72] L. W. Beineke and R. E. Pippert. A census of ball and disk dissections. *Lecture Notes in Mathematics*, 303:25–40, 1972.

[BP74] L. W. Beineke and R. E. Pippert. Enumerating dissectible polyhedra by their automorphism groups. *Canadian Journal of Mathematics*, 26(1):50–67, 1974.

[Bro63] W. G. Brown. Enumeration of non-separable planar maps. *Canadian Journal of Mathematics*, 15:526–545, 1963.

[Bro65] W. G. Brown. Historical note on a recurrent combinatorial problem. *American Mathematical Monthly*, 72:973–977, 1965.

[BT64] W. G. Brown and W. T. Tutte. On the enumeration of rooted non-separable planar maps. *Canadian Journal of Mathematics*, 16:572–577, 1964.

[Bur68] Yu. D. Burago. Neravenstva izoperimetricheskogo tipa v teorii poverchnostei ogranichennoi vneshnei krivizny (in russian). In *Sem in Mathematics*, volume 10, Leningrad, 1968. (Translation: Isoperimetric Inequalities in the Theory of surfaces of Bounded External Curvature, Consultant Bureau, New York- London, 1970).

[Cha94] P. Champ. Reverse engineering in industrial applications using laser stripe triangulation. In *3D Imaging and Analysis of Depth/Range Images, IEE Colloquium*, London, January 1994. 3D Scanners Ltd.

[CL96] B. Curless and M. Levoy. A volumetric method for building complex models from range image. In *Proc. of ACM SIGGRAPH'96*, pages 303–312, 1996.

[CW99] Y. H. Chen and Y. Z. Wang. Genetic algorithms for optimized re-triangulation in the context of reverse engineering. *Computer Aided Design*, 31:261–271, 1999.

[dC76] M. P. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice–Hall Inc., New Jersey, 1976.

[DHKL01] N. Dyn, K. Hormann, S.-J. Kim, and D. Levin. Optimizing 3D triangulations using discrete curvature analysis. In T. Lyche and L.L. Schumaker, editors, *Mathematical Methods for Curves and Surfaces: Oslo 2000*, Innovations in Applied Mathematics, pages 135–146. Vanderbilt University Press, Nashville, TN, 2001.

[Die99] J. Dieter. *Graph, Networks and Algorithms*. Springer, Berlin [etc.], 1999.

[DLR90] N. Dyn, D. Levin, and S. Rippa. Data dependent triangulations for piecewise linear interpolation. *IMA Journal on Numerical Analysis*, 10:137–154, 1990.

[DMSB02] M. P. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Discrete differential-geometry operators for triangulated 2–manifolds. In *Proc. of the International Workshop on Visualization and Mathematics (VisMath'02)*, page 27, Berlin, 2002.

[EM94] H. Edelsbrunner and E. Mücker. Three-dimensional α–shapes. *ACM Transactions on Graphics*, 13(1):43–72, 1994.

[Epp92] D. Eppstein. Approximating minimum weight triangulation. In *Proc. of the 3rd annual ACM-SIAM symposium on Discrete algorithms*, pages 48–57, Orlando, 1992.

[Eul59] L. Euler. Novi commentarii academiae scientiarium imperialis petropolitanae. 7:13–14, 1758–1759.

[Flo96] M. Floater. Mathematical technique for reverse engineering. SINTEF Report No. STF42 A96019, University of Oslo, Oslo, 1996.

[FMS03] I. Friedel, P. Mullen, and P. Schröder. Data–dependent fairing of subdivision surfaces. In *Proc. of the 8th ACM symposium on Solid modeling and applications*, pages 185–195, Seattle, 2003. ACM Press.

[FNP96] A. Ferko, Ľ. Niepel, and T. Plachetka. Criticism of hunting minimum weight triangualtion edges. In *Proc. of the 12th Spring Conference on Computer Graphics*, Bratislava, June 1996. http://www.cg.tuwien.ac.at/ wp/SCCG96-proceedings/

[FS91] B. Falcidieno and M. Spagnuolo. A new method for the characterization of topographic surfaces. *International Journal of Geographical information systems*, 5(4):397–412, 1991.

[Gar04] V. A. Garanzha. Variazionnyi method postroenija bilipshizevych parametrizazii negladkich poverchnostei (in russian). In *Proc. of Grid Generation: Theory and Applications*, pages 17–28, Moscow, June 2004. Russian Academy of Sciences. (Translation: Variational method of constructing bi–Lipschitz parametrizations of non–regular surfaces).

[GD02] Ch. Gold and M. Dakowicz. Terrain modelling based on contours and slopes. In D. Richardson, editor, *Advances in Spatial Data Handling: Proc. 10th Internationa Symposium Spatial Data Handling (SDH 2002)*. Springer–Verlag, 2002.

[Geb99] T. Gebäck. Improving triangulations for finite element analysis. Supervised by Martin Lindberg, August 1999.

[GMW97] M. Guo, J. Menon, and B. Willette. Surface reconstruction using α–shape. *Computer Graphics Forum*, 16(4):177–190, 1997.

[Guy58] R. K. Guy. Dissecting a polygon into triangles. *Bulletin of the Malayan Mathematical Society*, 5:57–60, 1958.

[HAP] Information Service HAPEG. $3d$ computed tomography as basic for reverse engineering. http://www.hapeg.de/skripte-eng/Formerfassung-eng.htm

[Har60] F. Harary. Unsolved problems in the enumeration of graphs. *Publications of the Mathematical institute of the Hungarian Academy of Sciences*, 5:1–20, 1960.

[Har68] F. Harary. *Graphical enumeration problems*, chapter I. Graph Theory and Theoretical Physics, pages 1–41. Academic Press, London, 1968.

[HDD+92] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics (Proc. of SIGGRAPH92)*, 26(2):71–78, 1992.

[KMST85] J. V. Knop, W. R. Müller, K. Szymanski, and N. Trinajstić. Computer generation of certain classes of molecules. *SKTH/Kemija u industriji, Zagreb*, page 253, 1985.

[Kon00a] E. V. Konstantinova. The constructive enumeration of square animals. In *Preprint at Com^2MaC, POSTECH*, volume 10, page 55, 2000.

[Kon00b] E. V. Konstantinova. The constructive enumeration of triangular animals. In *Preprint at Com^2MaC, POSTECH*, volume 21, page 33, 2000.

[Kon01] E. V. Konstantinova. Enumeration and generation animals. In *Report at Com^2MaC*, page 13, 2001.

[KST94] J. Köbler, U. Schöning, and J. Torán. *The graph isomorphism problem: its structural complexity.* Birkhauser Verlag, Basel, 1994.

[Kui70] N. H. Kuiper. Minimal total absolute curvature for immersions. *Inventiones Mathematicae*, 10:209–238, 1970.

[Law72] C. L. Lawson. Transforming triangulation. *Discrete Mathematics*, 3:365–372, 1972.

[LCK03] S. Lee, Y. Choi, and K. Kim. Shape reconstruction from unorganized points using voronoi diagrams. *The International Journal of Advanced Manufacturing Technology*, 21:446–451, 2003.

[Mas91] W. S. Massey. *A basic course in algebraic topology.* Springer–Verlag, Belrin, Hiedelberg, New York, 1991.

[MAT] The web's most extensive mathematics resource MATHWORLD. Catalan number.
http://mathworld.wolfram.com/CatalanNumber.html

[MD02] J.–L. Maltre and M. Daniel. Discrete curvatures and applications: a survey. Rapport de recherche No. 004.2002, Laboratore des Sciences de l'Information et des Systèmes (LSIS) ESIL, Marseille, 2002.

[MM63] J. W. Moon and L. Moser. Triangular dissections of n-gon. *Canadian Mathematical Bulletin*, 6:175–178, 1963.

[Moo67] W. Moon. Enumerating labelled tree. In F. Harary, editor, *Graph Theory and Theoretical Physics*, pages 261–271, London, New York, 1967. Academic Press.

[Mor] P. A. Morris. A catalog of trees on n nodes, $n < 14$. Mathematical Observations, Research and other Notes No. StA, Department of Mathematics, University of the West Indies (unpublished mimeograph).

[Mul65] R. C. Mullin. On counting triangular rooted maps. *Canadian Journal of Mathematics*, 17:373–382, 1965.

[Neg91] S. Negami. Diagonal flips of triangulations on surfaces. *Yokohama Mathematical Journal*, 47:1–40, 1991.

[O'R87] J. O'Rourke. Triangulation of minimum area as $3d$ object models. In *Proc. of the International Joint Conference on AI 81*, pages 664–666, 1987.

[oST] National Institute of Standarts and Technology. Dictionary of algorithms and data structures: Recursion. http://www.nist.gov/dads/HTML/recursion.html

[PB01] S. Petitjean and E. Boyer. Regular and non-regular point sets: properties and reconstruction. *Computational Geometry*, 19:101–126, 2001.

[PJB86] R. C. Jain P. J. Besl. Invariant surface characteristics for $3d$ object recognition in range images. *Computer Vision Graphics and Image Processing*, 33, 1986.

[QS90] E. Quak and L. Schumaker. Cubic spline fitting using data dependent triangulations. *Computer Aided Geometric Design*, 7:293–301, 1990.

[Rea72] R. C. Read. *Graph theory and computing*, chapter The coding of various kinds of unlabelled trees, pages 153–182. Academic Press, New York, 1972.

[RJPC02] H.-M. Rho, Y. Jun, S. Park, and H.-R. Choi. A rapid reverse engineering system for reproducing $3d$ human busts. *Annals of the CIRP*, 51:139–143, 2002.

[Sch02] J. R. Schewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry Theory & Applications*, 22(1–3):21–74, May 2002.

[Tak60] F. Takeo. On triangulated graph. In *I Bulletin of Fukuoka University of Education III*, volume 10, pages 9–21, 1960.

[Tak61] F. Takeo. On triangulated graph. In *II Bulletin of Fukuoka University of Education III*, volume 11, pages 17–31, 1961.

[Tak63] F. Takeo. On triangulated graph. In *III Bulletin of Fukuoka University of Education III*, volume 13, pages 11–21, 1963.

[Tak64] F. Takeo. On triangulated graph. In *IV Bulletin of Fukuoka University of Education III*, volume 14, pages 19–30, 1964.

[TC98] M. Teichmann and M. Capps. Surface reconstruction with anisotropic density-scaled alpha shapes. In *Proc. of IEEE Visualization*, pages 67–72, 1998.

[Tót04] Z. Tóth. Towards an optimal texture reconstruction. In *Proc. of the 8th Central European Seminar on Computer Graphics*, Wien, April 2004.
http://www.cg.tuwien.ac.at/studentwork/CESCG/CESCG-2004/

[Tri92] N. Trinajstić. *Chemical Graph Theory*. CRC Press, Bosa Raton, 2nd edition edition, 1992.

[Tru94] R. J. Trudeau. *Introduction to Graph Theory*. Dover Publications, Inc., New York, 1994.

[TSC+01] I. Trinks, S. Singh, C. Chapman, P. Barton, and M. Bosch. Traveltime tomography using irregular parameterised grids. In *LITHOS Science report*, pages 53–57, 2001.

[Tut62] W. T. Tutte. A census of planar triangulations. *Canadian Journal of Mathematics*, 14:21–38, 1962.

[Tut63] W. T. Tutte. A census of planar maps. *Canadian Journal of Mathematics*, 15:249–271, 1963.

[WW86] H. Wenchen and H. Wenjie. Generation and enumeration of planar polycyclic aromatic hydrocarbons. *Tetrahedron*, 42(19):5291–5299, 1986.

[XH03] X. Xu and K. Harada. Automatic surface reconstruction with α–shape method. *The Visual Computer*, 19:431–443, 2003.

Summary

In this thesis we deal with triangulations and try to solve the problem of constructing a triangulation of a scattered data set in $\mathbb{R}^3$. This an important technique in surface reconstruction and widely used in many applications, such as computer graphics and vision, mesh generation, terrain modelling, tomography, reverse engineering, pattern recognition, computational geometry, surface reconstruction, cartography, geology, stereology, architecture, medical imaging and many other fields.

The thesis consists of two parts. In part I we consider the construction of combinatorial triangulations of a sphere using notions from combinatorial theory, theory of groups and topological graph theory. In part II we consider triangulations constructed from real data, *i.e.,* polyhedral triangulated surfaces that span the given data. We deal with closed surfaces; which means that our triangulations represent polyhedral closed surfaces that are topologically equivalent to the **2D** sphere. These triangulations can also be viewed as geometrical realizations of combinatorial triangulations on the sphere.

Generating and enumerating a specific class of objects are fundamental problems in discrete mathematics, graph theory, computational geometry, the cell growth problem and many other fields. In part I we concentrate on the problem of combinatorially generating triangulations of the topological type of the sphere, which can be represented by planar graphs. We consider the construction of such triangulations without fixing the coordinates of points and develop different algorithms for generating all possible and all non-isomorphic triangulations for a given number of points with and without checking on isomorphisms. The problem of constructing triangulations without checking on isomorphisms has partially been solved: we are able to generate specific subclasses of triangulations without any control on isomorphisms.

We tried to explore various approaches. Some of these approaches were new, and some others inevitably turned out to be similar to the approaches known in literature. We give the corresponding references whenever it is appropriate.

Part I, besides being a report on our original research, can also be used as a survey of various combinatorial methods to construct triangulations.

In part II we concentrate on the problem of the geometrical construction of triangulations, by using data sets of points with fixed coordinates, and study the properties of the total absolute curvature and the mean curvature.
Firstly, we show that the mean criterion produces a better triangulation than the Tight criterion, even for a very "bad" initial triangulation. Then we develop a new technique, MEFA (multi–edge flipping algorithm), and experimentally show that it has a great advantage over EFA because, roughly speaking, MEFA is able to move from one local minimum to another one. MEFA has only one weak point: it performs flipping a number of edges per turn, but in certain cases this is not necessary. Finally, we develop the k–incremental algorithm, which in combination with MEFA and some optimality criterion, constructs an initial triangulation. The main idea of this algorithm is that at every step of the construction we add incrementally a number of points to a preceding triangulation and then apply MEFA with one of the optimality criteria: the Tight criterion or the mean criterion. As a result, the final triangulation of a given scattered data set, constructed by this algorithm, is an initial and optimal triangulation with respect to the chosen optimality criterion. The algorithm is not perfect yet and in our future work we are going to improve it.

ACKNOWLEDGMENTS

The 1st October 1999 I came to the Netherlands to start research at the faculty Applied Mathematics in the University of Twente. The result of this you can find in the present thesis. It could not been completed without guidance, support and encouragement of many people.
First of all, I would like to thank my promotor Prof. Dr. C. R. Traas and my daily supervisors Dr. L. Alboul and Dr. R. van Damme for entrusting me this interesting project and continuous guidance during this years. I appreciate all the effort and time they devoted helping me to reach the objective of the project. I am particular grateful to Dr. L. Alboul and Dr. R. van Damme who support and help me to finish this research and from whom I have learned a lot.
During my Ph.D. studies I was honored to be a member of NACM group. I would like to thank the current and former staff of this group for the pleasant and enjoyable company.
Before I came to the Netherlands I was living and studying in Russia. I would like to thank all my school and university teachers in Russia for providing me broad education.
I would also like to thank the members of the promotion commission for their interest in my research and discussions they prompted.
My life in the Netherlands would not be so comfortable without all my friends whom I met here and whom I had knew before I leaved Russia. Especially I would like to mention my former course–mate from the Novosibirsk State University, Arthur Iordanidi and Andrei Sleptchenko.
At last but not the least I want to thank my parents and my sister for their permanent support during all my life, without which it would not possible to attain anything.
It is very difficult to write an acknowledgement section as it is impossible to mention all the people you would like to thank. **Thank you all** who help me during my work and my life.

Alexandre Netchaev
1 September 2004
Enschede