University of Poitiers
Department XLIM-SIC

University of Seville
Department of Applied Maths I

Thesis submitted by Ana María PACHECO-MARTÍNEZ
for obtaining the degree of Doctor

Extracting cell complexes from 4–dimensional digital images

RÉSUMÉ

Une *image numérique* peut être définie comme un ensemble de n–xels sur une grille constituée de n–cubes. Les n–xels peuvent être identifiés avec : (1) les n–cubes de la grille, ou avec (2) les points centraux de ces n–cubes. Dans le premier cas, nous travaillons avec une *grille primale*, mais dans le deuxième cas, nous travaillons avec une *grille duale* construite à partir de la grille primale.

La *segmentation* consiste à calculer une partition d'une image en régions. Les n–xels ayant des caractéristiques similaires (couleur, intensité, etc.) sont regroupés. Schématiquement, à chaque n–xel est attribuée une étiquette, et chaque région de l'image est constituée de n–xels de même étiquette. En particulier, si les uniques étiquettes autorisées pour les n–xels sont "blanche" et "noire", la segmentation est dit *binaire* : les n–xels noirs forment le *premier plan* (*foreground*) ou région d'intérêt dans une tâche d'analyse d'image, et les n–xels blancs forment le *fond* (*background*).

Certains modèles, comme les *Graphes d'Adjacence de Régions* (RAGs), les *Graphes Duaux* (DGs) et la *carte topologique*, ont été proposés pour représenter les partitions en régions, et en particulier pour représenter la topologie de ces régions, c.a.d. les relations d'incidence et/ou d'adjacence entre les différentes régions. Le RAG [27] est un précurseur de ce type de modèles, et il a été une source d'inspiration des DGs [18] et de la carte topologique [9, 10]. Un RAG représente une image primale étiquetée par un graphe : les sommets du graphe correspondent à des régions de l'image, et les arêtes du graphe représentent les relations d'adjacence entre les régions. Les DGs sont un modèle permettant de résoudre certains inconvénients des RAGs pour représenter des images de dimension 2. La carte topologique est une extension des modèles précédents définie pour manipuler des images primales de dimension 2 et 3 ; elle représente non seulement les relations topologiques, mais aussi les relations géométriques.

D'autre part, les methodes "de type" *Marching cubes* [12, 23, 24] et *Kenmochi et al.* [14, 15, 16] construisent des complexes représentant la topologie de la région d'intérêt d'une image numérique binaire de dimension 3. Dans la première méthode, l'algorithme construit un complexe simplicial, dont 0–cellules sont des points des arêtes de la grille duale. Dans la deuxième méthode, les auteurs construisent un complexe cellulaire sur une grille duale, c.a.d les 0–cellules du complexe sont des sommets de la grille duale. Afin de construire le complexe, Kenmochi et al. calculent (à rotations près) les différentes configurations de sommets blancs et noirs d'un cube, puis, ils construisent les enveloppes convexes des points noirs de ces configurations. Ces enveloppes convexes définissent les cellules du complexe, à rotations près.

Le travail développé dans cette thèse étend la méthode de Kenmochi et al. en dimension 4. L'objectif est de construire un complexe cellulaire à partir d'une image numérique binaire définie sur une grille duale. Nous calculons d'abord les différentes configurations de sommets blancs et noirs d'un 4–cube (à isométries près), puis, nous construisons des enveloppes convexes définies par ces configurations. Ces enveloppes convexes sont construites par déformation du 4–cube d'origine, et nous distinguons différentes opérations de construction de base (déformation, dégénérescence de cellules, etc.) Enfin, nous précisons la construction du complexe cellulaire correspondant à une image duale par assemblage des cellules ainsi obtenues.

RESUMEN

Una *imagen digital* puede ser definida como un conjunto de n–xeles en un mallado constituido de n–cubos. Los n–xeles pueden ser identificados con: (1) los n–cubos del mallado, o con (2) los puntos centrales de estos n–cubos. En el primer caso, trabajamos con un *mallado primal*, mientras que en el segundo, trabajamos con un *mallado dual* construido a partir del mallado primal.

La *segmentación* consiste en calcular una partición de una imagen en regiones. Los n–xeles que tienen características similares (color, intensidad, etc.) son reagrupados. Esquemáticamente, a cada n–xel se le asocia una etiqueta, y cada región de la imagen está constituida de n–xeles con la misma etiqueta. En particular, si las únicas etiquetas permitidas para los n–xeles son "blanca" y "negra", la segmentación se dice *binaria*: los n–xeles negros forman el *primer plano* (*foreground*) o región de interés en cuestión de análisis de la imagen, y los n–xeles blancos forman el *fondo* (*background*).

Ciertos modelos, como los *Grafos de Adyacencia de Regiones* (RAGs), los *Grafos Duales* (DGs) y la *carta topológica*, han sido propuestos para representar las particiones en regiones, y en particular para representar la topología de estas regiones, es decir las relaciones de incidencia y/o adyacencia entre las diferentes regiones. El RAG [27] es un precursor de este tipo de modelos, y ha sido una fuente de inspiración de los DGs [18] y de la carta topológica [9, 10]. Un RAG representa una imagen primal etiquetada por un grafo: los vértices del grafo corresponden a regiones de la imagen, y las aristas del grafo representan las relaciones de adyacencia entre la regiones. Los DGs son un modelo que permite resolver ciertos inconvenientes de los RAGs para representar imágenes de dimensión 2. La carta topológica es una extensión de los modelos anteriores definida para manipular imágenes primales de dimensión 2 y 3, representando no solamente las relaciones topológicas, sino también las relaciones geométricas.

Por otra parte, los métodos "de tipo" *Marching cubes* [12, 23, 24] y *Kenmochi et al.* [14, 15, 16] construyen complejos que representan la topología de la región de interés de una imagen digital binaria de dimensión 3. En el primer método, el algoritmo construye un complejo simplicial cuyas 0–celdas son puntos de las aristas del mallado dual. En el segundo método, los autores construyen un complejo celular en un mallado dual, es decir las 0–celdas del complejo son vértices del mallado dual. Con el fin de construir el complejo, Kenmochi et al. calculan (salvo rotaciones) las diferentes configuraciones de vértices blancos y negros de un cubo, después, ellos construyen las envolventes convexas de los puntos negros de estas configuraciones. Estas envolventes convexas definen las celdas del complejo, salvo rotaciones.

El trabajo desarrollado en esta tesis extiende el método de Kenmochi et al. a dimensión 4. El objetivo es construir un complejo celular a partir de una imagen digital binaria definida en un mallado dual. Calculamos primero las diferentes configuraciones de vértices blancos y negros de un 4–cubo, salvo isometrías, después, construimos las envolventes convexas definidas por estas configuraciones. Estas envolventes convexas se construyen mediante deformaciones del 4–cubo de origen, para lo que distinguiremos distintas operaciones elementales de construcción (deformación, degeneración de celdas, etc.). Finalmente, precisamos la construcción del complejo correspondiente a una imagen dual ensamblando las celdas así obtenidas.

ACKNOWLEDGEMENTS

This thesis begins on September 30, 2007, when I leave my home town (Cádiz) in order to go to Sevilla for writing a doctoral thesis.

At Seville, during my master courses, I meet Prof. Núñez-Valdés (Juan), who will become tutor of this thesis. Juan shows me the "small world" of the research, and he welcomes me to his research group, allowing me to attend to my first conferences. Thanks to Juan, I meet several colleagues sharing my same interests. I have lived a lot of important moments of my "researcher life" with them. Thank you very much to all of them.

During the last months of the master, Juan introduces me to Prof. Real-Jurado (Pedro), one of the advisors of this thesis. Pedro helps me to get a grant (actually two grants), which allows me to continue researching. In mid-2008, I start with Pedro a research line that will become this thesis.

After obtaining my first grant, Pedro introduces me to Prof. Lienhardt (Pascal), who is also an advisor of this thesis. I will be eternally grateful to Pascal, because this thesis would never have existed without him. Pascal, every minute that I have spent working with you has been a pleasure and an honor for me. Thanks for helping me to overcome all the difficulties (as those related to language as those of intellectual type) that have emerged during these years.

I want to appreciate the help that my family has given me, not only during this thesis, but during all my life. I remember the first years on my way to the kindergarten, when I read the numbers of the houses; the wonderful years at the school; the choice of high school; and then that of the degree... Thanks for supporting my decisions, thanks for helping me to become the person I am today.

I am grateful to all my colleagues at the Department of Applied Mathematics I of Seville University. Thanks for giving me my first contract of employment, helping me to discover the "small world" of the teaching at the university level. Thank you very much for the support that you have given me during these years. Thanks for worrying about me, and for telling me the words I needed to hear.

I am also grateful to my colleagues at the Laboratory XLIM-SIC of Poitiers University. They have enabled that I have found that my three long stays (five months each of them) in Poitiers had been very quickly. Thanks for your professional and personal interest for me, thanks for your smiles, thanks for your kindness, and thanks for receiving me. You have enabled that I feel at home.

Finally, I am grateful to the members of the jury and to the reviewers who have drawn up the reports of this thesis.

CONTENTS

1. Introduction 1

2. Reconstruction methods 3
- 2.1 Context 3
- 2.2 State of the art 8
  - 2.2.1 Object reconstruction from labeled images 9
    - 2.2.1.1 Region Adjacency Graph (RAG) 9
    - 2.2.1.2 Dual Graphs (DGs) 10
    - 2.2.1.3 Topological map 12
  - 2.2.2 Object reconstruction from binary images 15
    - 2.2.2.1 Marching squares 15
    - 2.2.2.2 Kenmochi et al. method in $2D$ 16
    - 2.2.2.3 Marching squares vs. Kenmochi et al. method in $2D$ 16
    - 2.2.2.4 Marching cubes 18
    - 2.2.2.5 Kenmochi et al. method 26
    - 2.2.2.6 Marching cubes vs. Kenmochi et al. 31

3. Extracting 3–dimensional cell complexes from binary digital images 33
- 3.1 Look-up table construction 34
  - 3.1.1 Pattern subsets 34
    - 3.1.1.1 A first solution 34
    - 3.1.1.2 A second solution 44
  - 3.1.2 Pattern cells 50
    - 3.1.2.1 Finding the edges to degenerate 50
    - 3.1.2.2 An order relation on the edge degeneracies . . 55
    - 3.1.2.3 Convex hull of a pattern subset 58
- 3.2 Construction of the cell complex 64
  - 3.2.1 Subsets of points of an image 65
    - 3.2.1.1 Localizing subsets of points of an image . . . 65
    - 3.2.1.2 Classifying subsets of points of an image . . . 67
  - 3.2.2 Cells of the cell complex 68

3.2.2.1 Pattern cells of the cell complex 68
3.2.2.2 Inverting the isometries 69
3.3 Simplification of the cell complex 70
3.4 Comparison with Kenmochi et al. method 71

4. Extracting 4–dimensional cell complexes from binary digital images 75
4.1 Look-up table construction 75
4.1.1 Pattern subsets 75
4.1.1.1 A first solution 76
4.1.1.2 A second solution 83
4.1.2 Pattern cells 86
4.1.2.1 Convex hull of a pattern subset 88
4.2 Construction of the cell complex 94
4.2.1 Vertex level 94
4.2.2 Cell level 95
4.3 Simplification of the cell complex 96

5. Conclusions and future works 99

Bibliography 101

Appendix 104

A. Pattern subsets in $\mathbb{Z}^4$ 107

B. Pattern cells in $\mathbb{R}^4$ 129

LIST OF FIGURES

2.1 Photographs, statues and holograms are images 3
2.2 Approaches for defining a digital image 4
2.3 Primal and dual grid for $n = 2$ 4
2.4 Extruding a point, edge, square and cube 5
2.5 Primal and dual complexes 5
2.6 Neighborhoods between pixels 6
2.7 Neighborhoods between n–xels of the primal grid 7
2.8 Neighborhoods between n–xels of the dual grid 8
2.9 An image and its RAG 10
2.10 Different images with the same RAG 10
2.11 Multi-graph constructed from a RAG 11
2.12 Labeled image with three regions 12
2.13 A combinatorial map 13
2.14 Level 0 map 13
2.15 Level 1 map 14
2.16 Level 2 map 14
2.17 Level 3 map 15
2.18 Marching squares 16
2.19 Kenmochi et al. method in $2D$ 16
2.20 An example of grid in marching cubes 18
2.21 Unmarked and marked points of a grid in marching cubes . . . 20
2.22 Intersection points on a marching cubes grid 21
2.23 Surfaces intersecting complementary cubes 22
2.24 Portion of surface contained in the cubes 23
2.25 Ambiguous topologies 24
2.26 Holes in a face due to an ambiguity 25
2.27 3–dimensional binary digital image 29
2.28 Cell complex constructed from an image 30
2.29 Representation of an object 31

3.1 Steps of the procedure 33
3.2 Multi-graph associated with a subset of vertices of the unit cube 35
3.3 Isometry sending S into S' and C_S into $C_{S'}$ 36

3.4 Two isometries sending S into S' 36
3.5 Two isometries sending S into S' and C_S into $C_{S'}$ 37
3.6 Multi-graphs associated with the subsets of 7 vertices 40
3.7 Scheme of an efficient algorithm for finding pattern subsets . . 44
3.8 Rotation on a face of angle $\pi/2$, $-\pi/2$ and π radians 45
3.9 Rotation on an edge of angle π radians 45
3.10 Rotation on a vertex of angle $\frac{2}{3}\pi$ and $-\frac{2}{3}\pi$ radians 45
3.11 Reflection on a face 46
3.12 Reflection on an edge 46
3.13 Edge, face and volume degeneracy 50
3.14 Computing the convex hull of a subset of a cube 51
3.15 Degenerating any edge 52
3.16 Degenerating an edge of the cube 52
3.17 Decomposing a cube into two prisms 53
3.18 Degenerating an edge of the prism 53
3.19 Degenerating a pyramid from the apex 53
3.20 Deforming a pyramid from its base 54
3.21 A tetrahedron, triangle and edge degeneracy 54
3.22 Deforming a volume of 6 vertices into a volume of 5 vertices . 55
3.23 Importance of an order on the edge degeneracies 56
3.24 An order on the edge degeneracies 57
3.25 Rooted spanning trees in the cube 58
3.26 A square is deformed into a non-planar face with 4 vertices . . 59
3.27 A square is deformed into a triangle 60
3.28 A triangle is deformed into a triangle 60
3.29 Degeneracy of a triangle into an edge 61
3.30 Degeneracy of a pyramid into a square 61
3.31 Degeneracy of a tetrahedron into a triangle 61
3.32 Diagram of the procedure 65
3.33 Stages of the cube scanning 66
3.34 Pattern subsets of an image 68
3.35 Pattern cells of the cell complex 69
3.36 Attaching a square to a triangle 69
3.37 Cell complex constructed from an image 70
3.38 Simplified cell complex 71
3.39 Multi-graphs associated with configurations of voxels 72
3.40 Image comparison 73
3.41 Object comparison 74

4.1 Multi-graph associated with a subset of the unit 4–cube 76
4.2 Two non-isometric isomorphic multi-graphs 79

4.3 Multi-graphs associated with the subsets of 8 vertices 80
4.4 Degenerated 4–cube 86
4.5 Computing the convex hull of a subset of a 4–cube 87
4.6 Importance of an order on the edge degeneracies 87
4.7 Order on the degeneracy of edges of the 4–cube 88
4.8 A tetrahedron is deformed into a tetrahedron 89
4.9 A pyramid is deformed into a pyramid 89
4.10 A pyramid is deformed into a non-spatial volume 90
4.11 Non-spatial volume cut up into two cospatial volumes 91
4.12 Non-spatial volume cut up into two volumes 91
4.13 Degeneracy of a hypertetrahedron into a tetrahedron 92
4.14 A hypervolume is deformed into a cube with an inside edge . . 93
4.15 A hypervolume is deformed into two cospatial volumes 93

LIST OF TABLES

2.1 Extreme case of marching squares 17
2.2 Configurations with one (resp. seven) black voxel 19
2.3 Configurations of unmarked and marked vertices of a cube . . 19
2.4 Marching cubes look-up table 21
2.5 Configurations of unmarked and marked vertices of a square . 24
2.6 The ways a surface can intersect a square face 24
2.7 Neighborhoods resolving ambiguities 25
2.8 Configurations of voxels obtained by Kenmochi et al. 27
2.9 The 22 unit cells obtained by Kong and Roscoe 28
2.10 The 23 cells obtained by Kenmochi et al. 29
2.11 Chernyaev look-up table 31

3.1 Inner and outer square faces of a cube 39
3.2 Isomorphisms between some multi-graphs of 7 vertices 41
3.3 Pattern subsets in $\mathbb{Z}^3$ 43
3.4 CPU time for computing pattern subsets in $\mathbb{Z}^3$ 48
3.5 Pattern subsets by using lexicographic order 49
3.6 Pattern cells (together with its boundary) in $\mathbb{R}^3$ 64

4.1 Inner and outer cubic volumes of a 4–cube 78
4.2 Isomorphic and non-isomorphic multi-graphs with 8 vertices . 81
4.3 Pattern subsets in $\mathbb{Z}^4$ with 0,1,2,14,15,16 points 82
4.4 CPU time for computing pattern subsets in $\mathbb{Z}^4$ 85
4.5 Pattern subsets in $\mathbb{Z}^4$ with 0,1,2,14,15,16 points by using lexicographic order 85

1. INTRODUCTION

A *digital image* can be defined as a set of n–xels on a grid made up by n–cubes. The n–xels can be identified with (1) the n–cubes of the grid; or with (2) the central points of these n–cubes. In the first case, we work with a *primal grid*; whereas in the second one, we work with a *dual grid* constructed from the primal one.

Segmentation consists in computing a partition of an image into regions. The n–xels having similar characteristics (color, intensity, etc.) are regrouped. Schematically, each n–xel is assigned a label, and each region of the image is made up by n–xels with the same label. Particularly, if the only labels allowed for the n–xels are "white" and "black", the segmentation is said *binary*: the black n–xels form the *foreground* or region of interest in an image analysis task, and the white n–xels form the *background*.

Some models, as *Region Adjacency Graphs* (RAGs), *Dual Graphs* (DGs) and *Topological map*, have been proposed for representing partitions into regions, and particularly for representing the topology of these regions, i.e. the incidence and/or adjacency relations between the different regions. RAG [27] is a precursor of this type of models, and it has been a source of inspiration of the DGs [18] and the topological map [9, 10]. A RAG represents a labeled primal image by a graph: the vertices of the graph correspond to regions of the image, and the edges of the graph represent the adjacency relations between regions. DGs are a model allowing us to resolve some disadvantages of RAGs for representing 2–dimensional images. The topological map is an extension of the previous models defined for handling primal images of dimension 2 and 3, representing not only the topological relations but also the geometrical ones.

On the other hand, the methods "type" *Marching cubes* [12, 23, 24] and *Kenmochi et al.* [14, 15, 16] construct complexes representing the topology of the region of interest of a 3–dimensional binary digital image. In the first method, the algorithm constructs a simplicial complex, whose 0–cells are points of the edges of the dual grid. In the second one, the authors construct a cell complex on a dual grid, i.e. the 0–cells of the complex are vertices of the dual grid. In order to construct the complex, Kenmochi et al. compute (up to rotations) the different configurations of white and black vertices of a cube, and then, they construct the convex hulls of the black points of these configurations. These convex hulls define the cells of the complex, up to rotations.

The work developed in this thesis extends Kenmochi et al. method to dimension 4. The goal is to construct a cell complex from a binary digital image defined on a dual grid. First, we compute the different configurations of white and black vertices of a 4–cube, up to isometries, and then, we construct the convex hulls defined by these configurations. These convex hulls are constructed by deforming the original 4–cube, and we distinguish several basic construction operations (deformation, degeneracy of cells, etc.). Finally, we construct the cell complex corresponding to the dual image by assembling the cells so obtained.

This thesis is structured as follows. In Chapter 2, first, we recall some important notions related to digital images such as: primal and dual grid, primal and dual complex, neighborhood, etc.; and then, we list different methods for representing a partition of an image into regions, we show the advantages and disadvantages of these methods, and we compare each other. In Chapter 3, we construct a look-up table containing (up to isometries) all the cells which can be defined from a subset of vertices of a cube. These cells allow us to construct a cell complex associated with the image. Then, we simplify this complex in order to represent the only region of interest of the image. Finally, we compare the results obtained by using our method with those obtained by using Kenmochi et al. method. In Chapter 4, we extend to dimension 4 the results obtained in Chapter 3. Consequently, we construct a look-up table containing (up to isometries) all the cells which can be defined from a subset of vertices of a 4–cube. These cells allow us to construct a cell complex associated with the 4–dimensional image. Finally, the only region of interest of the image is obtained by simplifying this complex.

2. RECONSTRUCTION METHODS

2.1 Context

An *image* (from latin *imago*) is a representation of an object. It means an effective way of storing and communicating visual information, since images usually condensate or summary the information of the objects they represent.

Images of dimension two are, for instance, photographs whereas statues and holograms are examples of three-dimensional images. In Figure 2.1 several images are shown.

Fig. 2.1: Photographs are images of dimension two, whereas statues and holograms are images of dimension three.

Digitizing is a procedure for converting an image to numerical data. The image is divided into elements called *n–xels*, particularly *pixels* for $n = 2$ and *voxels* for $n = 3$. Every n–xel, localized in a position, is assigned a scalar value representing the brightness or darkness of the image at that point. *Digital images* are numerical representations of objects and they allow us to process images by using computational tools.

There does not exist a unique way of defining a digital image or of digitizing an image. This problem has been studied since 40 years and many different digital image definitions have been proposed. One can say that authors have followed two main approaches for defining digital images: (a) an analytical approach; and (b) a geometrical approach (see Figure 2.2 for examples of both approaches). From an analytical point of view, a gray-level digital image is a function from a subset $X \subset \mathbb{Z}^n$ to $\mathbb{R}$. Every $x \in X$

denotes the position of an n–xel, and $f(x)$ indicates the intensity (or gray level) of the image at that point. Moreover, this function can be represented by a matrix of size $s_1 \times ... \times s_n$, where s_k denotes the number of n–xels per unit length k. The element $i_1...i_n$ of the matrix coincides with $f(i_1, ..., i_n)$, i.e. it indicates the intensity of the n–xel localized in the position $(i_1, ..., i_n)$. Geometrically, a digital image is defined as a subset of a grid made up by n–cubes, particularly squares for $n = 2$ and cubes for $n = 3$.

$$\begin{pmatrix}
5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\
5 & 5 & 5 & 5 & 5 & 5 & 5 & 9 & 9 & 5 \\
5 & 9 & 9 & 9 & 5 & 5 & 5 & 5 & 5 & 5 \\
5 & 5 & 5 & 5 & 5 & 5 & 4 & 4 & 5 & 5 \\
5 & 5 & 5 & 5 & 5 & 4 & 4 & 4 & 4 & 5 \\
5 & 2 & 5 & 5 & 4 & 4 & 4 & 4 & 4 & 4 \\
2 & 2 & 2 & 5 & 5 & 8 & 8 & 8 & 8 & 5 \\
2 & 2 & 2 & 5 & 5 & 8 & 8 & 8 & 8 & 5 \\
5 & 1 & 5 & 5 & 5 & 8 & 8 & 6 & 8 & 5 \\
7 & 1 & 7 & 7 & 7 & 8 & 8 & 6 & 8 & 7
\end{pmatrix}$$

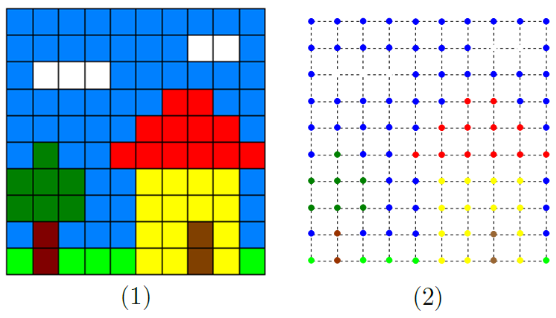


Fig. 2.2: On the left: 2–dimensional digital image defined by a matrix of size 10 pixels per width $\times$ 10 pixels per height. On the right: 2–dimensional digital image defined by (1) color squares and (2) color points, of a grid.

Focusing on the second approach, the n–xels of an image can be identified with (1) the n–cubes of the grid; or with (2) the central points of these n–cubes. In the first case, we work with a *primal grid* made up by n–cubes centered at points with integer coordinates; so that the corner points of these n–cubes are determined by points with semi-integer coordinates. In the second one, we work with a *dual grid* (constructed from the primal one) made up by n–cubes whose corner points are assigned to points with integer coordinates. The procedure for constructing the dual grid from a primal grid is shown in Figure 2.3 for $n = 2$.

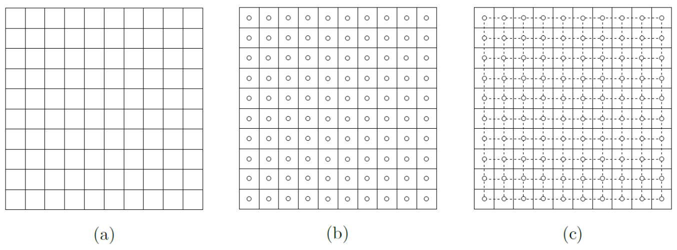


Fig. 2.3: By considering the central point (b) of each square of the primal grid (a), we construct a dual grid (c).

The digital images defined as subsets of primal grids can be considered as *cubical complexes* (see [7, 8]). These structures are similar to simplicial complexes but they are constructed from n–cubes instead of simplices. More concretely, each n–cube of a primal image is a unit subcomplex made up by 2^n vertices (0–cells), $n \cdot 2^{n-1}$ edges (1–cells) and generally, by $\binom{n}{k} \cdot 2^{n-k}$ k–faces (k–cells) with $0 \leq k \leq n-1$. The number of k–faces is obtained as follows: an n–cube is constructed by extruding an $(n-1)$–cube in any direction (see Figure 2.4), so the number of k–faces of an n–cube is twice the number of k–faces of an $(n-1)$–cube plus the number of $(k-1)$–faces of a $(n-1)$–cube. Moreover, the n–cube contains one n–cell which is its interior.

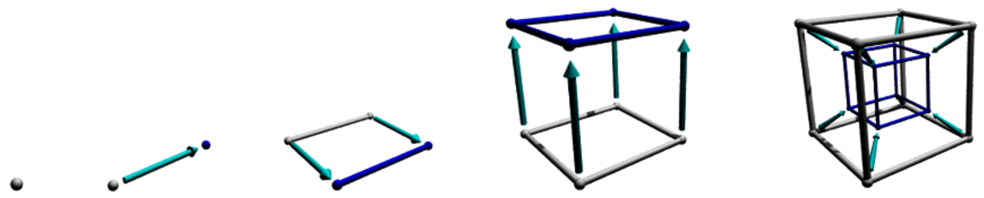


Fig. 2.4: By extruding a point, we construct an edge; by extruding an edge, we construct a square; by extruding a square, we construct a cube; and by extruding a cube, we construct a 4–cube.

The digital images defined as subsets of dual grids can be considered as *dual complexes* (see [19]). These complexes are constructed from primal complexes in the same way that dual grids are constructed from primal grids. More concretely, let I_P be a primal image and let $C(I_P)$ be the cubical complex associated with I_P. Every 0–cell of the dual complex $C^*(I_P)$ is defined by the central point of an n–cell of $C(I_P)$. A 1–cell of $C^*(I_P)$ is incident to two of its 0–cells if the two corresponding n–cells of $C(I_P)$ are incident to a same $(n-1)$–cell of it. More generally, every k–cell of $C^*(I_P)$ corresponds to a $(n-k)$–cell of $C(I_P)$ with $0 \leq k \leq n$. Figure 2.5 shows the construction of a dual complex from a cubical complex of dimension two.

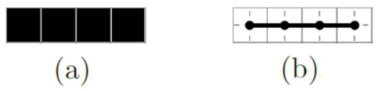


Fig. 2.5: (a) A 2–dimensional cubical complex, and (b) the corresponding dual complex.

Another important task in the context of digital images is to define *neighborhoods* between n–xels. These neighborhoods are determined by the adjacency relations between the n–xels. Moreover, these adjacency relations can be approached from two different points of view, depending on the considered grid: (a) one approach consists in determining the intersection of each pair of n–cubes of the primal grid; and (b) the other one consists in computing the *distance* between each pair of vertices of the dual grid. See Figure 2.6 for an example of both of them.

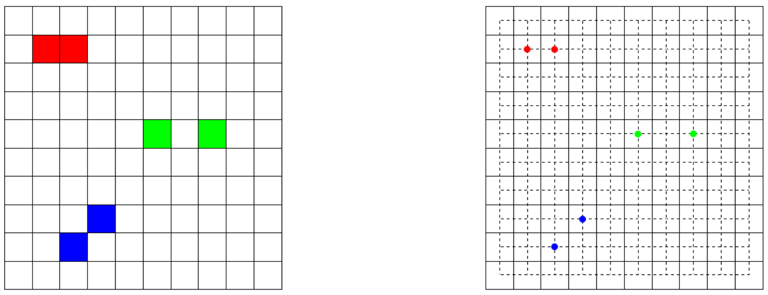


Fig. 2.6: On the left: the intersection of the two red pixels of the primal grid is an edge; that of the blue ones is a point; and that of the green ones is empty. On the right: the distance between the two red pixels of the dual grid is 1; that between the blue ones is $\sqrt{2}$; and that between the green ones is greater than $\sqrt{2}$, so they are not adjacent.

Taking into account that the non-empty intersection of two n–cubes of the primal grid is a k–cube with $0 \leq k \leq n-1$, following the first point of view, we have that two n–xels of the primal grid are *neighboring n–cubes* if they share a k–cube with $0 \leq k \leq n-1$. Depending on the values of k, different types of neighborhoods between n–xels can be defined. Particularly, (a) two pixels are *4–neighboring pixels* if they share one edge (1–cube), and they are *8–neighboring pixels* if they share at least one vertex (0–cube); (b) two voxels are *6–neighboring voxels* if they share one square face (2–cube), they are *18–neighboring voxels* if they share at least one edge, and they are *26–neighboring voxels* if they share at least one vertex; (c) two 4–xels are *8–neighboring 4–xels* if they share one cubic face (3–cube), they are *32–neighboring 4–xels* if they share at least one square face, they are *64–neighboring 4–xels* if they share at least one edge, and they are *80–neighboring 4–xels* if they share at least one vertex. A pictorial description about neighboring n–xels of the primal grid is shown in Figure 2.7.

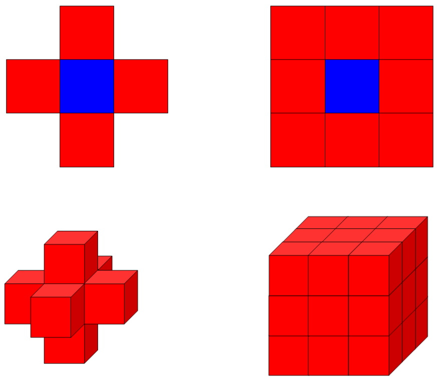


Fig. 2.7: On the top: a blue pixel of the primal grid with its 4 and 8–neighboring pixels (in red), respectively. On the bottom: a voxel of the primal grid with its 6 and 26–neighboring voxels (in red), respectively.

A *distance* is a function $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ satisfying the following properties: (a) $d(x, y) \geq 0$; (b) $d(x, y) = 0$ if and only if $x = y$; (c) $d(x, y) = d(y, x)$ and (d) $d(x, z) \leq d(x, y) + d(y, z)$.

Some well-known distances are: (1) $d_1(x, y) = \sum_{i=1}^{n} |x_i - y_i|$; (2) $d_2(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$; and (3) $d_\infty(x, y) = max_{i=1}^{n}\{|x_i - y_i|\}$.

Following the second point of view, the *neighborhood relation* between two vertices of the dual grid is related to their distance. Depending on the distance, different types of neighborhoods between n–xels of the dual grid can be obtained. Particularly, (a) two pixels $P = (i, j)$, $Q = (i', j')$ are *4–neighboring pixels* if they satisfy $d_1(P, Q) = |i - i'| + |j - j'| = 1$ and $d_\infty(P, Q) = max\{|i - i'|, |j - j'|\} = 1$, and they are *8–neighboring pixels* if they satisfy $d_1(P, Q) \leq 2$ and $d_\infty(P, Q) = 1$; (b) two voxels $P = (i, j, k)$, $Q = (i', j', k')$ are *6–neighboring voxels* if they satisfy $d_1(P, Q) = |i - i'| + |j - j'| + |k - k'| = 1$ and $d_\infty(P, Q) = max\{|i - i'|, |j - j'|, |k - k'|\} = 1$, they are *18–neighboring voxels* if they satisfy $d_1(P, Q) \leq 2$ and $d_\infty(P, Q) = 1$, and they are *26–neighboring voxels* if they satisfy $d_1(P, Q) \leq 3$ and $d_\infty(P, Q) = 1$; (c) two 4–xels $P = (i, j, k, l)$, $Q = (i', j', k', l')$ are *8–neighboring 4–xels* if they satisfy $d_1(P, Q) = |i - i'| + |j - j'| + |k - k'| + |l - l'| = 1$ and $d_\infty(P, Q) = max\{|i - i'|, |j - j'|, |k - k'|, |l - l'|\} = 1$, they are *32–nieghboring 4–xels* if they satisfy $d_1(P, Q) \leq 2$ and $d_\infty(P, Q) = 1$, they are *64–neighboring 4–xels* if they satisfy $d_1(P, Q) \leq 3$ and $d_\infty(P, Q) = 1$, and they are *80–neighboring 4–xels* if they satisfy $d_1(P, Q) \leq 4$ and $d_\infty(P, Q) = 1$. A pictorial description about neighboring n–xels of the dual grid is shown in Figure 2.8.

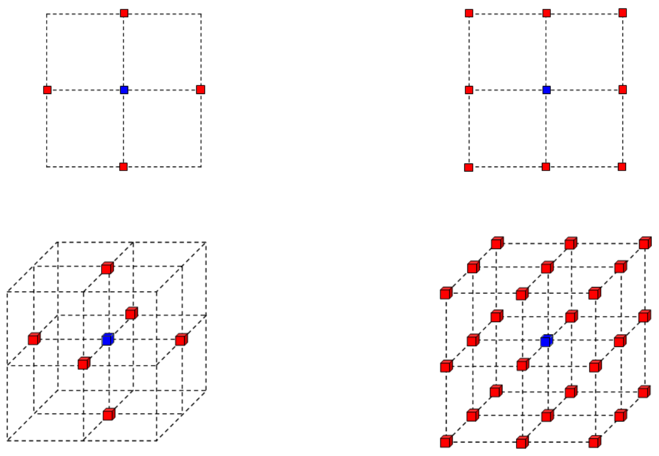


Fig. 2.8: On the top: a blue pixel of the dual grid with its 4 and 8–neighboring pixels (in red), respectively. On the bottom: a blue voxel of the dual grid with its 6 and 26–neighboring voxels (in red), respectively.

More questions about the relationship between both approaches of neighboring n–xels are studied in [11].

2.2 State of the art

As we have commented previously, from a geometrical point of view, a digital image is a set of n–xels with a color. An image is said *labeled* if every n–xel is assigned with a label. Particularly, if the only labels allowed for the n–xels are "white" and "black", the image is said *binary*. A *region* of an image is defined as a maximal connected set of n–xels with the same label. Moreover, if the image is a binary one there exist only two regions: the black n–xels form the *foreground* or region of interest in an image analysis task; and the white n–xels form the *background*. On the other hand, if more than two labels are allowed then the image has multiple regions, and each region can be considered as a binary image, in such a way that the n–xels of the region form the foreground and the remaining of the image forms the background.

Several authors [10, 15, 18, 23, 27] have developed some methods for reconstructing objects from digital images. They represent a partition of an image into regions, and they study the incidence and adjacency relations between regions. Particularly, in the case of binary images, there exists only one region of interest, so it suffices to represent the boundary of it.

In Section 2.2.1, we describe several methods which construct different "models" from labeled images. These "models" represent the incidence and/or adjacency relations between the different regions of an image. As a precursor of this type of methods, we present the *Region Adjacency Graph* (RAG) [27], source of inspiration of the *Dual Graphs* (DGs) [18] and the *topological map* [9, 10]. A RAG represents a labeled primal image by a graph. DGs are a method allowing us to resolve some disadvantages of RAGs for representing 2–dimensional images. The topological map is an extension of the previous methods defined for handling primal images of any dimension, representing not only the geometrical relations but also the topological ones.

In Section 2.2.2, we introduce some methods defined for handling binary digital images. These methods construct complexes representing the topology of the region of interest of a 3–dimensional binary digital image. In this sense, we highlight the methods "type" *marching cubes* [12, 23, 24] and *Kenmochi et al.* [14, 15, 16]. In the first method, the algorithm constructs a simplicial complex whose 0–cells are points of the edges of the dual grid. In the second one, the authors construct a cell complex on a dual grid, i.e. the 0–cells of the complex are vertices of the dual grid. At the end of this section, we show that Kenmochi et al. method can be considered as an extreme case of marching cubes.

2.2.1 Object reconstruction from labeled images

The methods shown in this section use as input data labeled digital images placed on a primal grid. These images contain multiple regions, so they cannot only be represented by their boundary.

2.2.1.1 Region Adjacency Graph (RAG)

In [27], Rosenfeld introduces the first method for representing 2–dimensional images with multiple regions placed on a primal grid. An image of this type is made up by a subset of color squares of the grid satisfying that all the squares of a connected region have the same color, different from the colors of the adjacent regions. In this sense, two regions are *adjacent* if (a) the color of the squares of one of them is different from the color of the squares of the other one, and (b) there exists at least a curve bordering both regions.

The image is represented by a graph whose vertices are the regions of the image and whose edges join the pairs of adjacent regions. This graph is called *Region Adjacency Graph (RAG)*. The RAG provides a "simple-connectivity view" of the image. Figure 2.9 shows an example of a 2–dimensional image with seven regions and its corresponding RAG.

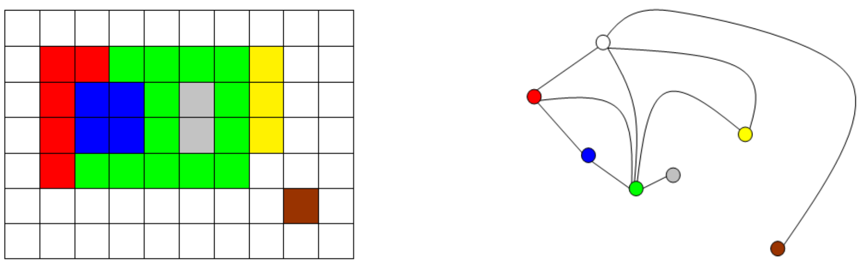


Fig. 2.9: On the left: 2–dimensional digital image with seven regions placed on a primal grid. On the right: RAG consisting of seven vertices and nine edges which show the adjacency relations between the regions of the image.

The two main advantages of RAGs are: (a) they are easy to define; and (b) they can be extended easily to higher dimension. On the other hand, the disadvantages of RAGs are: (a) there is no difference between inclusion and adjacency relations (see grey region and green region in Figure 2.9); (b) they do not represent multiple adjacency, for instance, in Figure 2.9 the red region is twice adjacent to the green region; and (c) two images that are not topologically equivalent can have the same RAG (see Figure 2.10). Many of these disadvantages are resolved in dimension two by using the *dual graphs*.

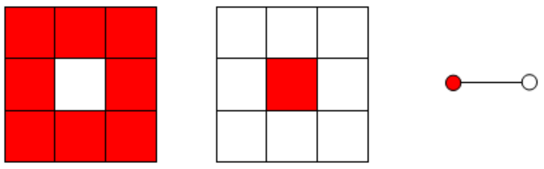


Fig. 2.10: Two images topologically non-equivalent with the same RAG.

2.2.1.2 Dual Graphs (DGs)

In [18], Kropastch introduces the dual graphs. These graphs extend the notion of RAG. The *dual graphs* consist of two graphs: (1) a multi-graph, which extends the RAG by adding the inclusion relations and multiple adjacency between the regions; and (2) its dual graph.

The multi-graph inherits the vertices and the edges of the RAG; it attaches loops for describing the inclusion relations between regions; and it adds edges between the pairs of vertices which represent multi-adjacent regions. More concretely, each loop describing an inclusion relation is defined on the vertex which represents the container region and it encloses the vertex which represents the contained region; and the number of edges between each pair of vertices coincides with the number of times the corresponding regions are adjacent. Figure 2.11 (a) shows the multi-graph corresponding to the image which appears in Figure 2.9.

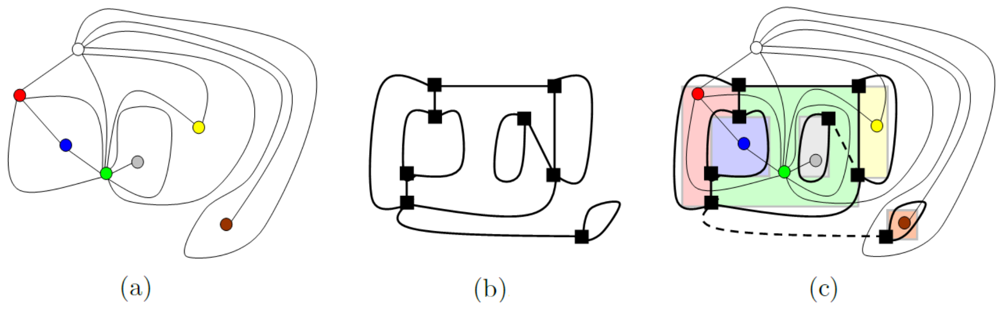


Fig. 2.11: (a) Multi-graph constructed from the RAG shown in Figure 2.9. (b) Its dual graph. (c) Both of them representing the image.

The dual graph of this multi-graph is a graph (indeed a multi-graph) and it is constructed as follows: vertices of the dual graph are associated with intersection points of maximal curves dividing the two same regions, and edges (up to isthmuses) of the dual graph correspond to curves bordering two adjacent regions. Moreover, each isthmus between two vertices of the dual graph corresponds to a loop of the multi-graph constructed from the RAG. In Figure 2.11 (b), we show the dual graph of the multi-graph shown in Figure 2.11 (a).

Remark 2.2.1. *Let us note that the multi-graph constructed from the RAG represents the dual of the image, since for each region of the image there is a vertex of the multi-graph, and for each curve bordering two adjacent regions there is an edge of the multi-graph. Consequently, the dual graph of this multi-graph represents the dual of the dual of the image, i.e. this dual graph is equivalent to the primal image.*

The image is represented by both the multi-graph constructed from the RAG and its graph dual. Figure 2.11 (c) shows the representation of the image in Figure 2.9 by using dual graphs. Let us note that the dual graph of the multi-graph has two “fictive” edges (dotted lines) joining the boundary of the white region with that of the brown region, and the boundary of the green region with that of the grey region, respectively. These edges are “fictive” because they do not represent a curve bordering two adjacent regions, although they are necessary to preserve the connection of the dual graph. The dual of each of these “fictive” edges on the multi-graph is the loop around the brown region, and the loop around the grey region, respectively. These loops allow us to differentiate between adjacency relations and inclusion relations.

DGs resolve many disadvantages of RAGs, but they have other disadvantages: (a) two graphs are necessary to represent the image; (b) there exist "fictive" edges that do not represent curves bordering two adjacent regions; and (c) they cannot be extended easily to higher dimension. In this sense, Damiand et al. works are addressed to improve and extend RAGs and DGs for representing higher dimension images.

2.2.1.3 Topological map

Damiand et al. define 3–dimensional digital images as subsets of a primal grid (see, for instance, [9, 10]). The voxels of these images are represented by cubes of the primal grid. They work with labeled images, so two voxels with the same label belong to the same connected region (see Figure 2.12 extracted of [10]).

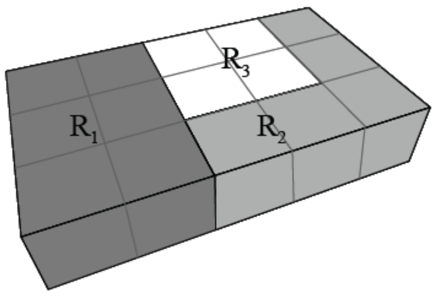


Fig. 2.12: Labeled image with three regions: R_1 has six voxels, R_2 has five voxels, and R_3 has four voxels.

They define the *topological map* as a model which represents both the topological and geometrical information of a three-dimensional labeled image.

First, Damiand et al. construct a model that describes all the geometric relations between the elements of the image. This model is called *level 0 map* and it is constructed as follows: the image is considered as a cubical complex, where each voxel is a cube made up by eight 0–cells, twelve 1–cells, and six 2–cells together with the incidence relations between them. More concretely, a 1–cell is incident to two 0–cells and a 2–cell is incident to four 1–cells. These voxel subdivisions together with the incidence relations between the cells are represented by a *combinatorial map* (see [22]).

The concept of combinatorial map generalizes the notion of graph since it not only describes the incidence relations between 0–cells and 1–cells, but it also describes the whole topological information. A *combinatorial map* is defined by a set of abstract elements called *darts* and by a set of permutations (some of them being involutions). A dart corresponds to a n–cell seen from a $(n-1)$–cell seen from a $(n-2)$–cell ... seen from a 0–cell. Permutations link these abstract elements for describing the cells. See Figure 2.13 for an example of combinatorial map.

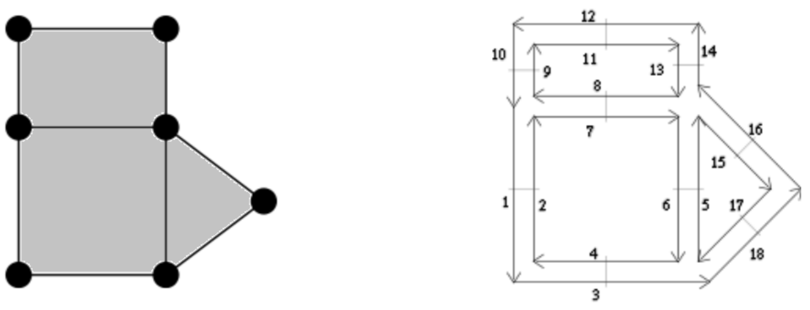


Fig. 2.13: The combinatorial map on the right represents the subdivision on the left. Darts are represented by numbered arrows. Moreover, two darts linked by the composition of the permutation and the involution are drawn consecutively (for instance, darts {1} and {3}); and two darts linked by the involution are drawn parallel and in reverse orientation (for instance, darts {7} and {8}). More concretely, the darts {15},{16},{17},{18} describe the apex of the triangle; darts {5},{6} describe its base; and darts $\{\{5\},\{6\},\{15\},\{16\},\{17\},\{18\}\}$ describe the triangle.

The level 0 map of the image shown in Figure 2.12 appears in Figure 2.14.

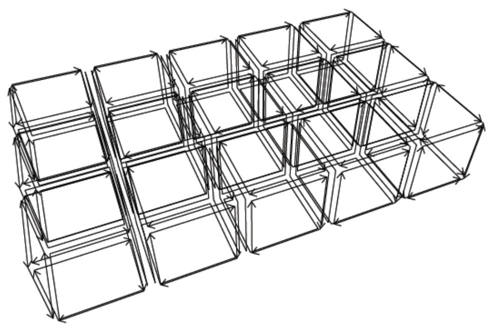


Fig. 2.14: Level 0 map of the labeled image shown in Figure 2.12.

Second, they construct the *level 1 map.* This map is obtained from the level 0 map by removing each face between two voxels having the same label; it allows us to merge these pairs of voxels. These faces are removed because they correspond to inner faces of the same region. Consequently, the non-removed faces border different regions. In Figure 2.15, the level 1 map of the image shown in Figure 2.12 is presented.

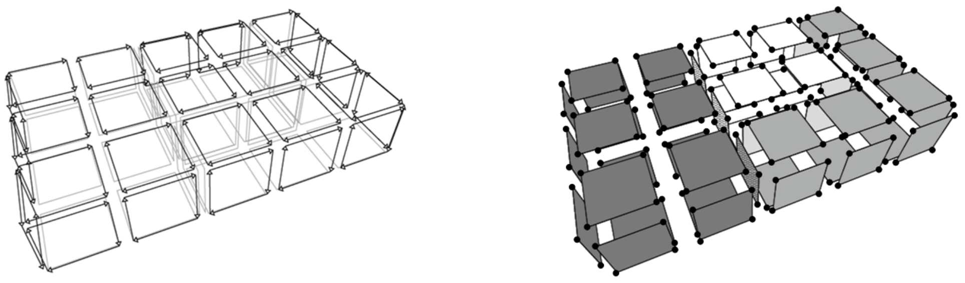


Fig. 2.15: On the left: level 1 map obtained from the level 0 map shown in Figure 2.14. On the right: subdivision corresponding to the level 1 map.

The following step in Damiand et al. method is to construct the *level 2 map*. This map is obtained from the level 1 map by removing the degree two edges, i.e. the edges incident to exactly two faces of voxels having the same label; it allows us to merge these pairs of faces. Figure 2.16 presents the level 2 map of the image shown in Figure 2.12.

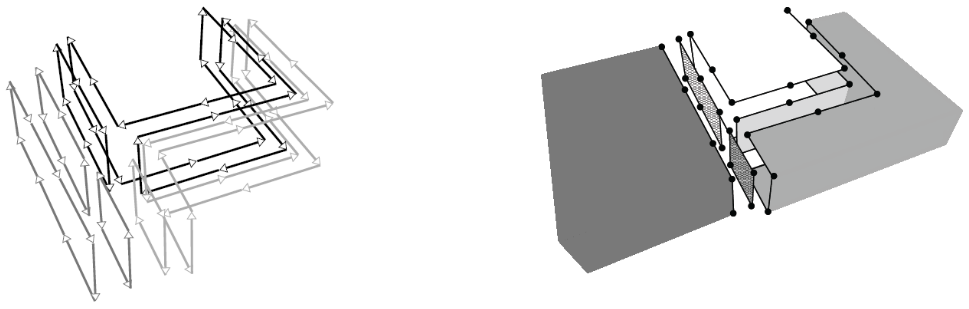


Fig. 2.16: On the left: level 2 map obtained from the level 1 map shown in Figure 2.15. On the right: subdivision corresponding to the level 2 map.

The minimal topological information of the image is represented from the *level 3 map*. This map is constructed from level 2 map by removing the degree two vertices, i.e. the vertices incident to exactly two edges of voxels having the same label; it allows us to merge these pairs of edges. The level 3 map is the minimal map (in number of cells) which describes the topology of the object, since it is not possible to remove any cell without losing topological information. In Figure 2.17, we present the level 3 map of the image shown in Figure 2.12.

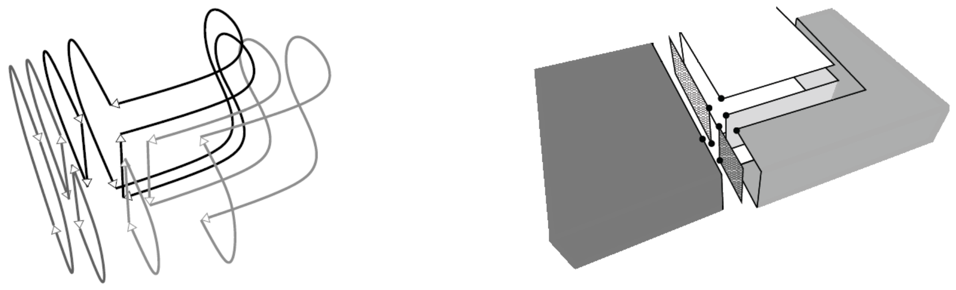


Fig. 2.17: On the left: level 3 map obtained from the level 2 map shown in Figure 2.16. On the right: geometrical subdivision of the image shown in Figure 2.12, corresponding to the level 3 map.

Summarizing, the topological map obtained from the image is defined by three levels: (a) removal of faces; (b) removal of edges; and (c) removal of vertices.

2.2.2 Object reconstruction from binary images

The methods shown in this section, on the contrary to the previous ones, use as input data binary digital images placed on a dual grid. Here the images contain only one region of interest, so they can be represented by their boundary.

2.2.2.1 Marching squares

Given a subset of black points in a grid made up by squares (see Figure 2.18 (a)), the algorithm constructs a simplicial complex (see Figure 2.18 (b)) which approximates the curve containing the points (see Figure 2.18 (c)). The edges of the squares of the grid which have one black end-point and the other one white are intersected by the curve containing the points. Moreover, these intersection points (shown in red in Figure 2.18 (b)) are the 0–simplices of the complex. The last step in marching squares is to construct the 1–simplices of the complex from the 0–simplices.

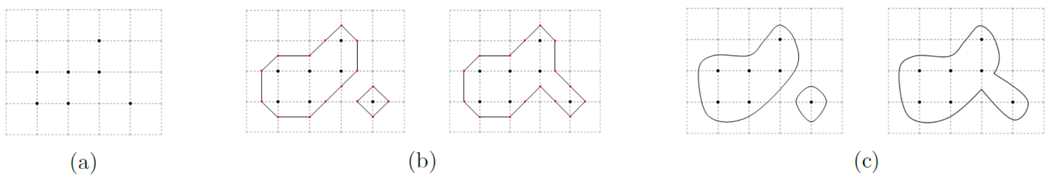


Fig. 2.18: (a) Subset of points; (b) simplicial complexes extracted from (a); and (c) curves containing the points in (a).

2.2.2.2 Kenmochi et al. method in 2D

Given a subset of black points in a grid made up by squares, Kenmochi et al. method constructs a cell complex from the convex hulls of the black vertices of each square. The vertices, edges and faces of these convex hulls are the cells of dimension 0, 1 and 2, respectively, of the cell complex (see Figure 2.19).

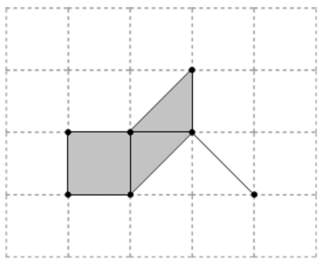
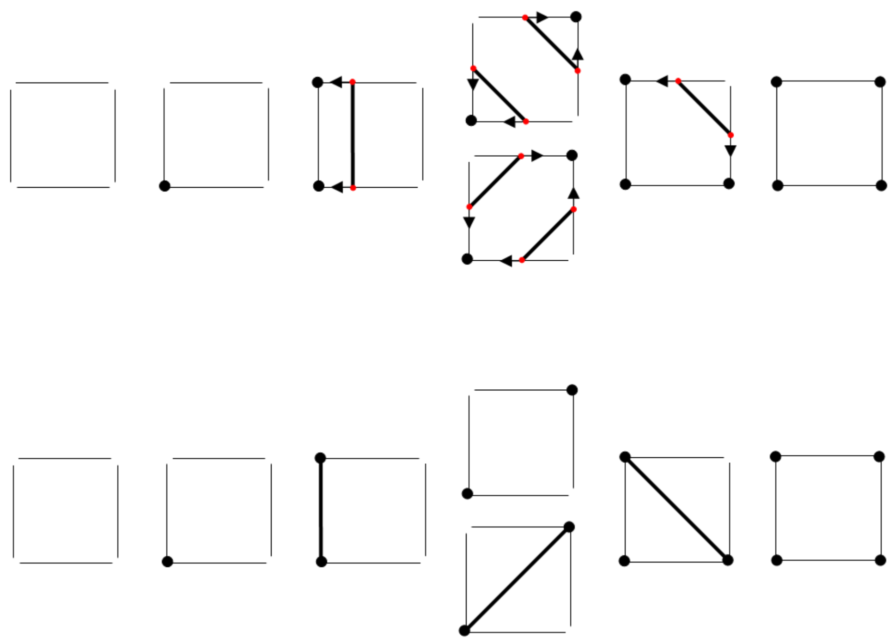


Fig. 2.19: Cell complex constructed from Figure 2.18 (a).

2.2.2.3 Marching squares vs. Kenmochi et al. method in 2D

We consider the extreme case of marching squares in which the black points of the grid coincide with the intersection points between the edges of the grid and the curve containing the black points (see Table 2.1).



Tab. 2.1: The portion of the simplicial complex contained in each square of the grid corresponding to the extreme case (on the bottom) can be determined by "sending" the intersection points (shown in red) to the corresponding black vertices of the square.

Let us observe that the results shown on the bottom in Table 2.1 coincide with the boundary of the convex hulls of the black vertices, except for the case where there are both two diagonally opposite black vertices and two diagonally opposite white vertices. In this case there are ambiguous topologies. This ambiguity can be resolved in a similar way that proposed by Lachaud et al. in [21], i.e. by choosing a correct neighborhood between the black vertices and other one between the white vertices. In dimension two, the different pairs of neighborhoods between black vertices and white vertices are $\{(4,4),(4,8),(8,4),(8,8)\}$. A study of these pairs allows us to set down that by choosing the pair (4,8) or (8,4), the intersection of the curve with any square is univocally determined. Moreover, by choosing the 8–neighborhood between the black vertices and the 4–neighborhood between the white vertices, each of the portions of the complex obtained in this extreme case coincides with the boundary of one of the portions of the complex constructed by Kenmochi et al.

Remark 2.2.2. *Let us observe that the pair* (λ, μ) *of neighborhood relations for white and black vertices of the square, respectively, is called* Jordan pair*.*

Remark 2.2.3. *If in this extreme case of marching squares we choose a neighborhood relation more restrictive between white vertices than between black vertices, then the simplicial complex constructed by marching squares coincide with the boundary of the cell complex obtained by Kenmochi et al.*

2.2.2.4 Marching cubes

The method of marching cubes, developed by Lorensen and Cline in 1987 [23], uses as input data a scalar value α and a subset of points in $\mathbb{R}^3$ placed on a grid and associated with scalar values (see Figure 2.20 for an example). The basic idea of marching cubes is to suppose that the subset of points together with their assigned scalar values approximate a scalar function f on $\mathbb{R}^3$. The goal of this method is to construct a simplicial complex approximating the surface defined by $f(x, y, z) = \alpha$.

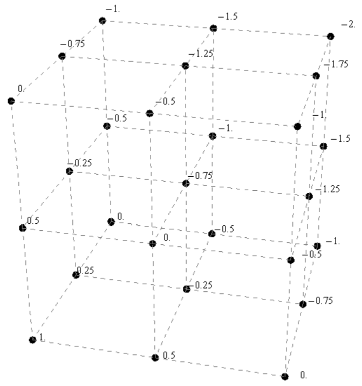
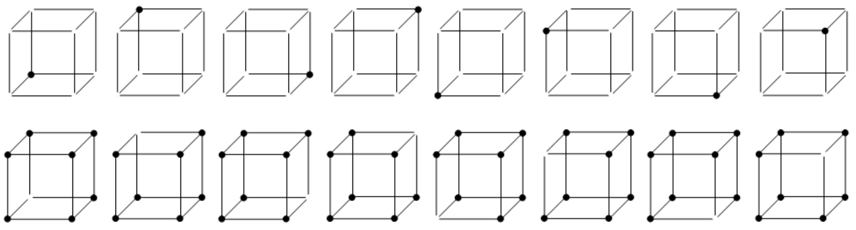


Fig. 2.20: Grid obtained from the subset of points $\{(0,0,0), (0,0,0.5), (0,0,1), (0,0.5,0), (0,0.5,0.5), (0,0.5,1), (0,1,0), (0,1,0.5), (0,1,1), (0.5,0,0), (0.5,0,0.5), (0.5,0,1), (0.5,0.5,0), (0.5,0.5,0.5), (0.5,0.5,1), (0.5,1,0), (0.5,1,0.5), (0.5,1,1), (1,0,0), (1,0,0.5), (1,0,1), (1,0.5,0), (1,0.5,0.5), (1,0.5,1), (1,1,0), (1,1,0.5), (1,1,1)\}$, whose assigned values are $\{0, -0.5, -1, -0.5, -1, -1.5, -1, -1.5, -2, 0.25, -0.25, -0.75, -0.25, -0.75, -1.25, -0.75, -1.25, -1.75, 1, 0.5, 0, 0.5, 0, -0.5, 0, -0.5, -1\}$, respectively.

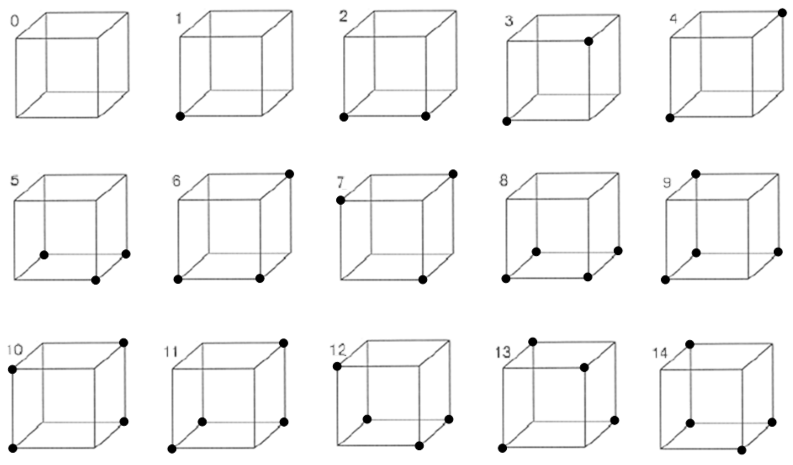
First, the algorithm processes the vertices of the cubes of the grid marking those vertices whose assigned value is equal to or greater than α. The vertices of the cubes whose value is lesser than α are unmarked.

A cube has eight vertices and each vertex can be a marked vertex or an unmarked vertex, so there exist $2^8 = 256$ configurations of unmarked and marked vertices of a cube. More precisely, there exist $C(8, c)$ configurations with c marked vertices and $8 - c$ unmarked vertices of a cube. Properties of combinatorial numbers allows us to set down that $C(8, 8 - c) = C(8, c)$, so that $C(8, c)$ is also the number of configurations with $8 - c$ marked vertices and c unmarked vertices of a cube. In Table 2.2 it is shown an example for $c = 1$.



Tab. 2.2: On the top: configurations with one marked vertex and seven unmarked vertices of a cube. On the bottom: configurations with seven marked vertices and one unmarked vertex of a cube.

Marching cubes considers that two configurations on a cube which have the same number of marked vertices are identical if there exists a rotation which sends the first configuration to the second one. Hence, by ignoring complementary cases derived from the properties of combinatorial numbers, marching cubes reduces the 256 configurations of unmarked and marked vertices of a cube to 23 configurations (15 configurations with $0 \leq c \leq 4$ marked vertices, and 8 complementary configurations with $5 \leq c \leq 8$ marked vertices). In Table 2.3, the 15 configurations with $0 \leq c \leq 4$ marked vertices of a cube are shown.



Tab. 2.3: Configurations of unmarked and marked vertices of a cube obtained by Lorensen and Cline.

In their works, they state that any subset of points in $\mathbb{R}^3$ placed on a grid and associated with scalar values is made up by combining (up to complementary cases and rotations) these 15 configurations of unmarked and marked vertices, taking into account that it is not necessary to use all them and any of them can be used more than once. In Figure 2.21, we present the result obtained by applying the first step of marching cubes algorithm to the set of points shown in Figure 2.20 for $\alpha = -0.1$.

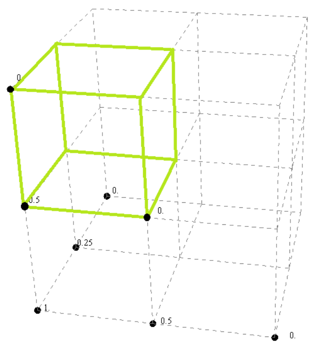


Fig. 2.21: The points with value equal to or greater than -0.1 are marked; whereas the points with values lesser than -0.1 are unmarked. Each one of the 8 configurations of unmarked and marked vertices coincides (up to complementary cases and rotations) with one of the configurations shown in Table 2.3. More concretely, the configuration on the green cube coincides with the Case 5 in Table 2.3.

Second, the edges of the cubes which have one marked end-point v_m and the other one unmarked v_u are closed intervals on $\mathbb{R}^3$ satisfying $f(v_m) - \alpha \geq 0$ and $f(v_u) - \alpha < 0$; so that Bolzano's Theorem allows us to set down that the surface $f(x, y, z) = \alpha$ intersects these edges. Moreover, the intersection point of the surface with each of these edges can be estimated by using linear interpolation. More concretely, if e is an edge of one of the cubes of the grid made up by a marked vertex $v_m(e)$ and a unmarked vertex $v_u(e)$, whose assigned values are $v_{v_m(e)}$ and $v_{v_u(e)}$, respectively, then the intersection point of the surface $f(x, y, z) = \alpha$ with the edge e is given by $v_m(e) + \frac{\alpha - v_{v_m(e)}}{v_{v_u(e)} - v_{v_m(e)}}(v_u(e) - v_m(e))$. In this way, the intersection points of the surface with the cubes of the grid are estimated.

In Figure 2.22, we show the intersection points of the surface $f(x, y, z) = -0.1$ with the cubes of the grid shown in Figure 2.20. The computations have been made by using the method described above.

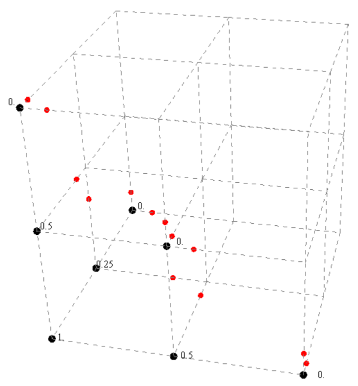
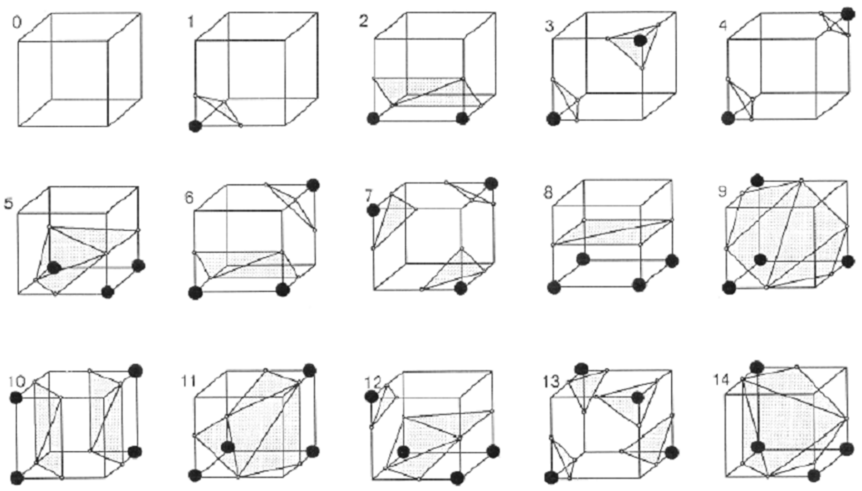


Fig. 2.22: In red, intersection points (computed by using linear interpolation) of the surface $f(x, y, z) = -0.1$ with the edges of the cubes of the grid shown in Figure 2.20.

The last step in marching cubes is to generate triangular facets representing the portion of surface that intersects each of the cubes. In Table 2.4, Lorensen and Cline present a triangulation of each of the 15 configurations of unmarked and marked vertices shown in Table 2.3.



Tab. 2.4: The ways a surface can intersect a cube obtained by Lorensen and Cline in [23].

Remark 2.2.4. *Let us note that the portions of surface shown in Table 2.4 correspond to choose a neighborhood relation more restrictive between the marked vertices than between the unmarked vertices. Moreover, let X be one of the configurations shown in Table 2.3 and let X' be the complementary* *configuration; if on the cube containing to X' we choose a neighborhood relation more restrictive between the unmarked vertices than between the marked vertices, then the portion of surface intersecting this cube is the same than the portion of surface intersecting the cube containing to X. See Figure 2.23 for an example.*

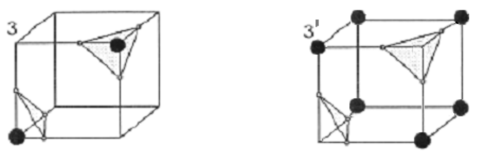


Fig. 2.23: On the left: the portion of surface corresponding to the Case 3 in Table 2.3. On the right: the portion of surface corresponding to the complementary configuration by choosing a neighborhood relation more restrictive between the unmarked vertices than between the marked vertices.

In their works, they state that the approximation of any surface is made up by combining (up to complementary cases and rotations) the 15 ways a surface can intersect a cube shown in Table 2.4, taking into account that it is not necessary to use all them and any of them can be used more than once. In Figure 2.24, we show a triangulation of the portion of surface contained in each of the cubes of the grid shown in Figure 2.20, as well as the approximation of the triangulated surface defined by $f(x, y, z) = -0.1$.

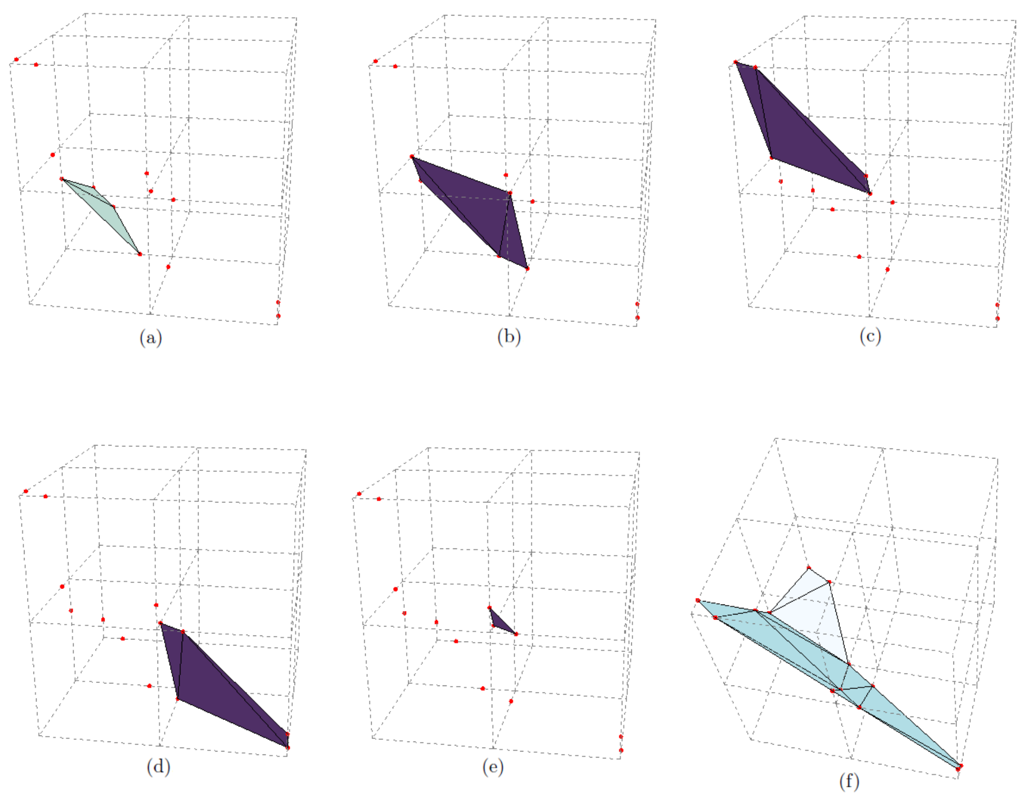


Fig. 2.24: The pictures (a), (b), (c), (d), (e) coincide (up to rotations) with the Cases 2, 5, 5, 5, 1 in Table 2.4, respectively, and they show the portion of surface contained in each of the cubes of the grid; the picture (f) shows the approximation of the triangulated surface.

Marching cubes problems

About seven years after marching cubes method was introduced, several authors [24, 28] demonstrated that the portion of surface that intersects each of the cubes (see Table 2.4) is not univocally determined by the intersection points (see Figure 2.25 for an example), i.e. there are several surfaces which can intersect the cubes in those intersection points. More concretely, the previous authors obtained seven *ambiguous topologies* corresponding to the Cases 3, 4, 6, 7, 10, 12 and 13 in Table 2.4, respectively.

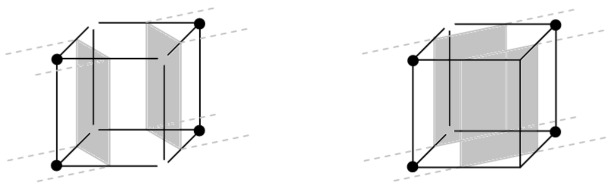
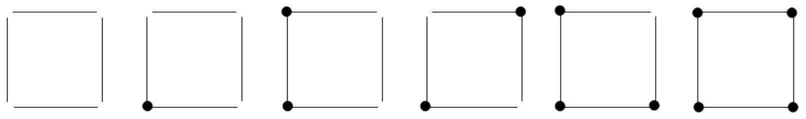


Fig. 2.25: The intersection points obtained by using linear interpolation give rise to both surfaces and it is not possible to take a decision on the interpretation of these types of situations.

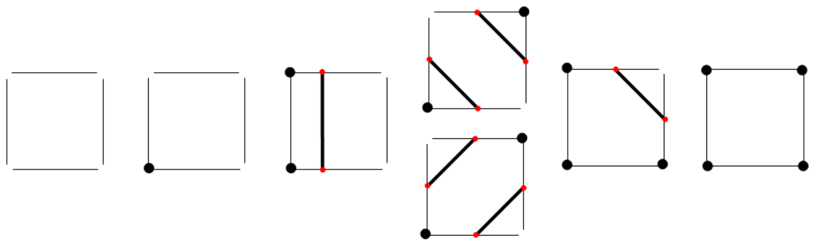
In [21], Lachaud et al. propose a solution for lifting these ambiguities. The idea is to choose a correct neighborhood relation between the marked vertices and other one between the unmarked vertices of the cube.

First, Lachaud et al. consider the different configurations of unmarked and marked vertices of a face of a cube (see Table 2.5).



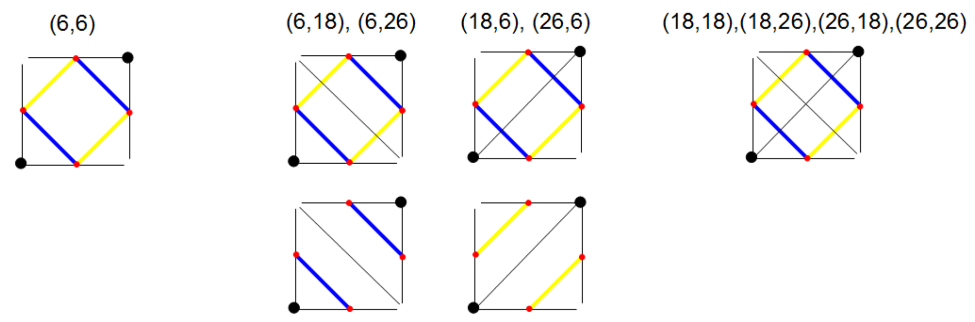
Tab. 2.5: Configurations of unmarked and marked vertices of a square face.

Marching cubes algorithm determines the intersection points of the surface with the edges having a marked vertex and a unmarked vertex. Table 2.6 shows the ways a surface can intersect a face of a cube.



Tab. 2.6: The ways a surface can intersect a square face.

Let us observe that the intersection of the surface with each face is univocally determined, except for the case where there are both two diagonally opposite marked vertices and two diagonally opposite unmarked vertices. This ambiguity can be resolved by choosing a correct neighborhood between the vertices marked and other one between the unmarked vertices. In dimension three, the different pairs of neighborhoods between marked vertices and unmarked vertices are $\{(6,6),(6,18),(6,26),(18,6),(18,18),(18,26),(26,6),(26,18),(26,26)\}$. In Table 2.7, we show the intersections of the surface with a face with two diagonally opposite marked vertices and two diagonally opposite unmarked vertices by using each of these pairs.



Tab. 2.7: The blue (resp. yellow) segments show a neighborhood relation more restrictive between the marked (resp. unmarked) vertices. The 6–neighborhood between marked vertices and the 6–neighborhood between unmarked vertices does not allow to resolve the ambiguity; and neither do the (18,18), (18,26), (26,18), (26,26). However, the (6,18), (6,26), (18,6) and (26,6) allow us to resolve it.

Remark 2.2.5. *Let us observe that the pair* (λ,μ) *of neighborhood relations for unmarked and marked vertices of the faces of a cube, respectively, is called* Jordan pair.

The study of the different pairs of neighborhoods allows us to set down that by choosing the pair (6,18), (6,26), (18,6), or (26,6), the intersection of the surface with any face of a cube is univocally determined.

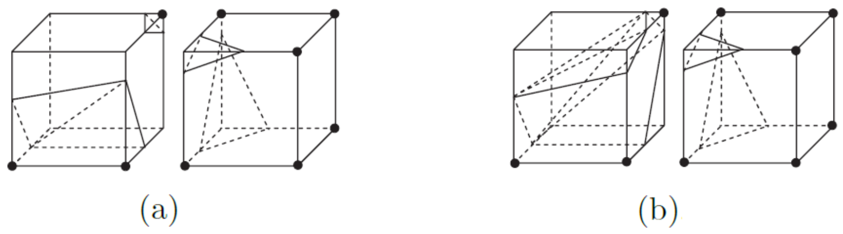


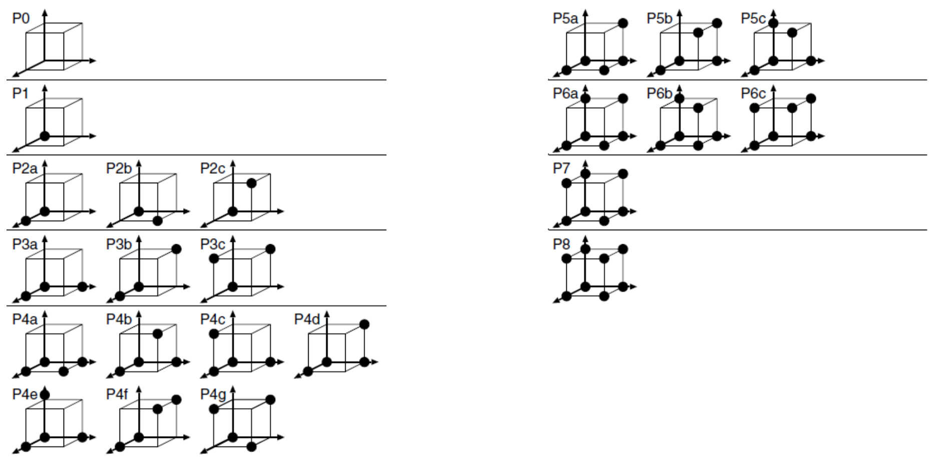
Fig. 2.26: (a) Surface with holes due to an ambiguity in a face shared by two cubes. (b) Choosing the 26–neighborhood between the marked vertices, and the 6–neighborhood between the unmarked vertices resolves the inconsistency.

Marching cubes can also produce a *topologically inconsistent* surface that contains holes caused by the ambiguities. Figure 2.26 (a) (extracted of [25]) shows two adjacent cubes which share a face. The portion of surface contained in each of these two cubes corresponds to the Cases 6 and 3 in Table 2.4, respectively. In the figure, the shared face is intersected differently in each cube. This ambiguity produces a hole in the surface. The inconsistency can be resolved by choosing the same pair of neighborhoods for each cube, in which case, we obtain that the shared face is intersected in the same way in each cube. For instance, in Figure 2.26 (a) the inconsistency can resolved by choosing the 26–neighborhood between the marked vertices, and the 6–neighborhood between the unmarked vertices (see Figure 2.26 (b)).

2.2.2.5 Kenmochi et al. method

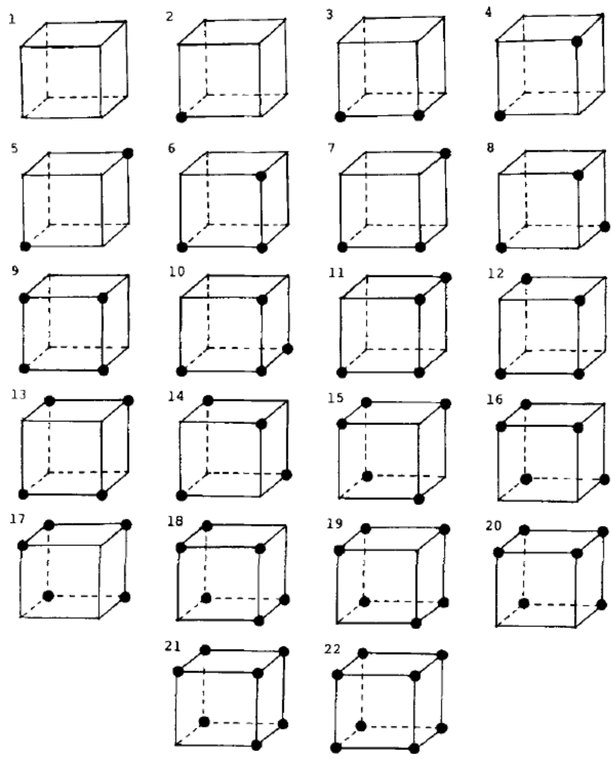
Kenmochi et al. define 3–dimensional digital images as subsets of a dual grid (see, for instance, [14, 15, 16]). The voxels of these images are the vertices of the cubes of the dual grid. They work with binary images, so that the voxels are white or black. Kenmochi et al. construct a *cell complex* from the black voxels of the image, in such a way that the boundary of the complex represents the object. The *cell complex* consists of a collection of *cells* together with the information on how they are attached to each other. The black voxels are the 0–cells of the complex. Every cell is defined by the black vertices of a cube of the grid. More concretely, every cell is an open set including the points inside of the convex hull of the vertices which define it. Moreover, the boundary of the cell is given by the vertices, edges and faces of the convex hull. The cells are attached to each other along their boundaries.

The same reasoning used in marching cubes allows us to set down that there exist $2^8 = 256$ configurations of white and black voxels on a cube. By considering the notion of identical configurations in the same sense as marching cubes, Kenmochi et al. reduce the 256 configurations of white and black voxels on a cube to 23 configurations. In Table 2.8, extracted of [14], these 23 configurations of white and black voxels on a cube are shown.



Tab. 2.8: Configurations of white and black voxels on a cube obtained by Kenmochi et al.

Remark 2.2.6. *Let us note that similar results were obtained in 1985 by Kong and Roscoe [17], who used Pólya's enumeration theorem (see [5]) for computing the number of different ways in which it can color the corner points of a cube by using two colors (black and white) taking into account that two colorings are the same if one is a rotation of the other. In this way, they obtained 23 configurations, and they constructed Table 2.9, where the only remaining configuration corresponds to a reflection of the cell 11. Hence, they determined 22* unit cells *(configurations of white and black voxels on a cube together with the edges, the faces and the volume of the cube).*



Tab. 2.9: The 22 unit cells obtained by Kong and Roscoe (extracted of [17]).

In their works, Kenmochi et al. state that any 3–dimensional binary digital image is made up by combining (up to rotations) the 23 configurations of white and black voxels shown in Table 2.8, taking into account that it is not necessary to use all them and any of them can be used more than once. In this way, given a 3–dimensional binary digital image on a dual grid, every configuration of white and black voxels on a cube of the grid coincides (up to rotations) with one of the 23 configurations shown in Table 2.8. In Figure 2.27, we show an example of a 3–dimensional binary digital image which consists of 39 black voxels on a dual grid of size $4 \times 4 \times 4$.

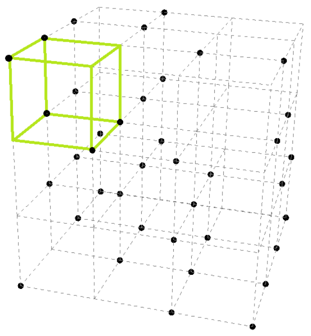
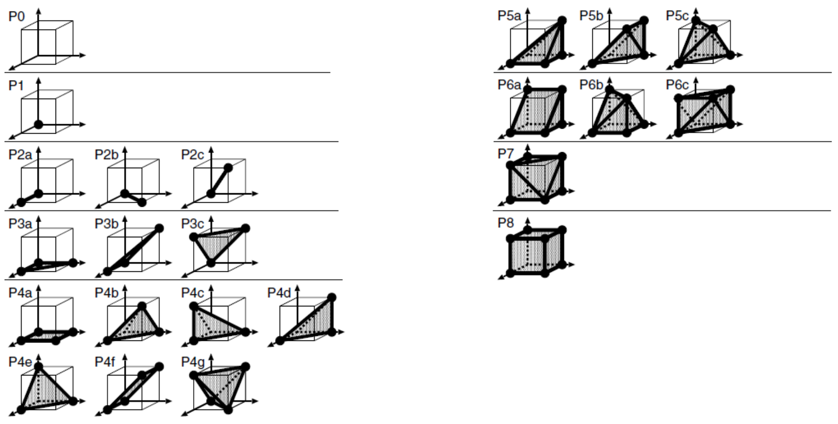


Fig. 2.27: Each one of the 27 configurations coincides (up to rotations) with one of the configurations in Table 2.8. More concretely, the configuration on the green cube coincides (up to rotations) with the configuration P5b in Table 2.8.

Next, for each configuration in Table 2.8, Kenmochi et al. construct a cell defined by the black voxels of the configuration. We recall that each of these cells is an open set including the points inside of the convex hull of the vertices which define it, and the boundary of the cell is given by the vertices, edges and faces of the convex hull. In this way, Kenmochi et al. obtain the 23 cells shown in Table 2.10 from the black voxels of the configurations shown in Table 2.8. These cells are used later for constructing the cell complex whose boundary represents the object encoded by the image.



Tab. 2.10: The 23 cells obtained by Kenmochi et al. (extracted of [14]) from the black voxels of the configurations shown in Table 2.8.

Remark 2.2.7. *A similar work was described in [17] for determining the "continuous analog" of the 22 unit cells. This structure is defined by Kong and Roscoe as a union of points, edges, triangles and tetrahedra obtained from the corner points of every unit cell.*

In their works, Kenmochi et al. state that the cell complex corresponding to a 3–dimensional binary digital image is made up by combining (up to rotations) the 23 cells shown in Table 2.10. In this way, every cell of the complex coincides (up to rotations) with one of the 23 cells shown in Table 2.10. In Figure 2.28, it is shown the cell complex corresponding to the image in Figure 2.27.

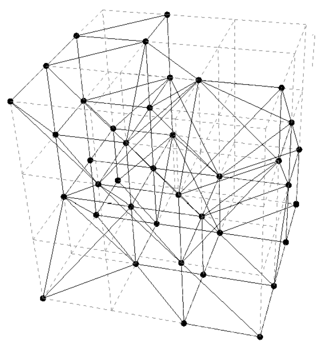


Fig. 2.28: Cell complex constructed from the 3–dimensional binary digital image shown in Figure 2.27 by using Kenmochi et al. method.

Finally, Kenmochi et al. extract the boundary of the complex. This boundary includes all the information, since the image is a binary one. Therefore, the object is contained in only one region and its internal structure is not necessary to represent it. In this sense, they represent the object as the set of 2–cells (together with their boundary) incident to exactly one of the 3–cells of the complex constructed from the image. More concretely, they make a list of all the 2–cells incident to at least one of these 3–cells. The set of 2–cells which represents the object is chosen from this list by counting the number of times each 2–cell appears in the list. If the 2–cell appears more than once, it is incident to two 3–cells, so it is not in the representation of the object since it is localized inside of it; whereas if the 2–cell appears only once, it is incident to exactly one of the 3–cells, so it is in the representation of the object since it is in the boundary of it. In Figure 2.29, we show the boundary of the cell complex in Figure 2.28.

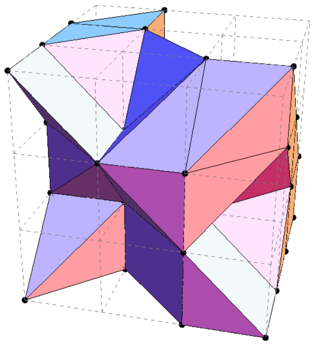
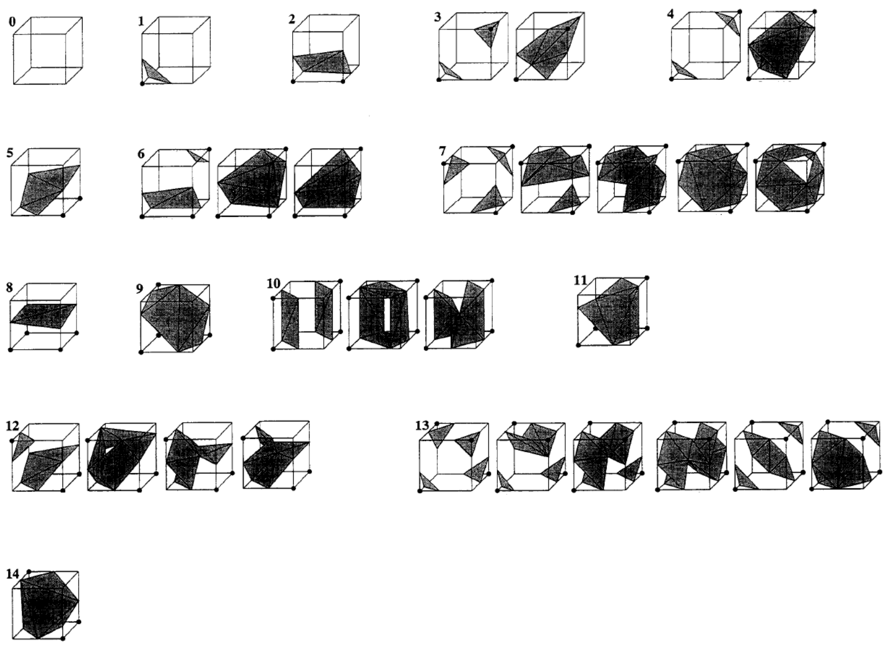


Fig. 2.29: Representation of the object encoded by the digital image shown in Figure 2.27 by using Kenmochi et al. method.

2.2.2.6 Marching cubes vs. Kenmochi et al.

By reasoning in a similar way as dimension two, Kenmochi et al. method in dimension three can also be considered as an extreme case of marching cubes.



Tab. 2.11: The ways a surface can intersect a cube obtained by Chernyaev in [6].

Let us recall that from Table 2.3 marching cubes algorithm computes the intersection points of the surface $f(x, y, z) = \alpha$ with the edges of each of the cubes whose end-points are a marked vertex and an unmarked vertex. In

1995, Chernyaev [6] showed the 33 portions of surface that can be obtained from these points (see Table 2.11).

In an analogous way as two-dimensional case, in the extreme case of all the marked vertices are assigned the value α, i.e. the surface $f(x, y, z) = \alpha$ crosses all the marked vertices, the intersection points coincide with the marked vertices. Moreover, the portions of surface corresponding to this extreme case can be determined by "sending" the intersection points obtained in Table 2.11 to the corresponding marked vertices of the cube.

Likewise, the portions of surface corresponding to this extreme case coincide (up to complementary cases) with the boundary of the cells shown in Table 2.10, except for the cases where there are ambiguous topologies, i.e. the Cases 3, 4, 6, 7, 10, 12 and 13. As we have commented previously, these ambiguities can be resolved by choosing one of these four pairs of neighborhoods: (6,18), (6,26), (18,6), or (26,6). Moreover, by choosing one of the last two pairs, each of the portions of surface obtained in this extreme case coincides with the boundary of one of the cells in Table 2.10.

Remark 2.2.8. *Analogously to dimension 2, the pair (λ, μ) of neighborhood relations for unmarked and marked vertices of the cube, respectively, is called* Jordan pair*.*

Remark 2.2.9. *If in this extreme case of marching cubes we choose a neighborhood relation more restrictive between unmarked vertices than between marked vertices, then the simplicial complex constructed by marching cubes coincide with the boundary of the cell complex obtained by Kenmochi et al.*

3. EXTRACTING 3–DIMENSIONAL CELL COMPLEXES FROM BINARY DIGITAL IMAGES

In this chapter we retrieve the works [14, 15, 16] developed by Kenmochi et al., in which topological tools are used to construct cell complexes from 3–dimensional binary digital images. We recall that Kenmochi et al. define the binary digital images as subsets of white and black points on a dual grid. The black points are the 0–cells of a cell complex. Here, every cell of the complex is constructed by using an algorithmic procedure which consists in *deforming* a cube of the dual grid, in such a way that the cell is defined by the black points of the cube. This deformation is a consequence of *degenerating*[1] edges of the cube incident to white vertices. The cell complex is formed by attaching the cells along their boundaries. See Figure 3.1 for an example. Finally, we simplify the complex by keeping the 0,1,2–cells which are not incident to a 3–cell, together with the 2–cells (and their boundary) which are incident to exactly one 3–cell of the complex.

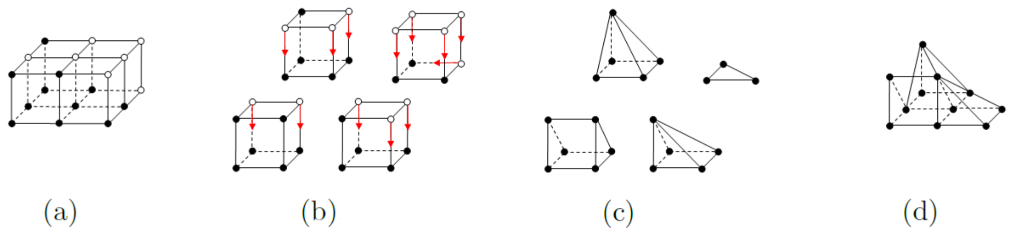


Fig. 3.1: (a) Binary image on a dual grid of size $3 \times 3 \times 2$; (b) deformation of each cube of the grid by degenerating edges incident to white vertices; (c) cells of the cell complex; and (d) cell complex corresponding to the image in (a).

The outline of this chapter is as follows: in Section 3.1, we construct a look-up table containing (up to isometries) all the cells which can be obtained by deforming a cube; in Section 3.2, we present a procedure for constructing the cell complex; in Section 3.3, we conceive an algorithm for simplifying the

1. deforming a k–cell into a $(k-1)$–cell belonging to the boundary of the k–cell

complex; and finally, in Section 3.4, we compare our results with the results of Kenmochi et al.

3.1 Look-up table construction

We construct all the cells which can be obtained (up to isometries) by deforming a cube. The procedure is done in two steps: (a) we determine the vertices which define these cells, and we store them in a look-up table; and (b) we compute the convex hull of these vertices. Each cell is an open set made up by all the points inside of the convex hull of the vertices which define the cell. The boundary of this convex hull is given by its vertices, edges, and faces. The cells (together with their boundary) are stored in a look-up table shown at the end of this section.

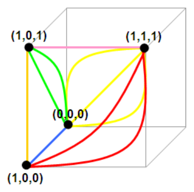
3.1.1 Pattern subsets

We compute the different sets of vertices defining the cells which can be obtained (up to isometries) by deforming a cube. The set of vertices which defines a cell is the set of black points of a cube of the grid. In Chapter 2, we exposed that there exist $2^8 = 256$ subsets of points which can be constructed from the vertices of a cube. The points of each subset are black vertices of the cube, whereas the points of the complement subset are white vertices of the cube. Next, we explain two methods for computing the isometric subsets, and we store a representative of each isometry class. They are the *pattern subsets*.

3.1.1.1 A first solution

There exist several solutions for finding these pattern subsets. A first idea, easy to implement, consists in associating each subset with a multi-graph, in such a way that two subsets identified with a same pattern subset are associated with isomorphic multi-graphs.

Below, we describe the method for constructing these multi-graphs. Let $S = \{p_1, ..., p_c\}$ be a subset with $1 \leq c \leq 8$ points. The vertices of the multi-graph G_S associated with S are the points $p_1, ..., p_c$; and the number of edges between each pair of vertices $p_i, p_j \in G_S$ is determined by the number of different coordinates between p_i and p_j. This way of constructing the edges of G_S takes into account the relative positions of the vertices in the space. In Figure 3.2, we show the multi-graph associated with the subset $S = \{(0,0,0), (1,0,0), (1,0,1), (1,1,1)\}$ of the unit cube.



$$\begin{array}{c|cccc} & (0,0,0) & (1,0,0) & (1,0,1) & (1,1,1) \\ \hline (0,0,0) & 0 & 1 & 2 & 3 \\ (1,0,0) & 1 & 0 & 1 & 2 \\ (1,0,1) & 2 & 1 & 0 & 1 \\ (1,1,1) & 3 & 2 & 1 & 0 \end{array}$$

Fig. 3.2: The vertices of the multi-graph associated with $S = \{(0,0,0),(1,0,0),(1,0,1),(1,1,1)\}$ are the points of S and the element m_{ij} of its adjacency matrix coincides with the number of different coordinates between the vertices p_i and p_j.

Considering the previous association between multi-graphs and subsets, it is natural to establish Definition 3.1.1.

Definition 3.1.1. *Let S and S' be two subsets of vertices of a cube. We say that S and S' are* isomorphic subsets *if their respective associated multi-graphs are isomorphic.*

In Theorem 3.1.2, we show that two subsets with at least 4 points are isomorphic iff they are identified with a same pattern subset.

Theorem 3.1.2. *Let S and S' be two subsets with $4 \leq c \leq 8$ points. S and S' are isomorphic subsets if and only if there exists a linear isometry $f : \mathbb{R}^3 \to \mathbb{R}^3$ which sends S into S' and the cube C_S containing S into the cube $C_{S'}$ containing S'.*

Remark 3.1.3. *Let us note that the condition of sending C_S into $C_{S'}$ leads to the fact that the isometry also sends the subset $C_S - \{S\}$ into the subset $C_{S'} - \{S'\}$.*

Proof. The scheme of the proof is the following.

We construct an isometry from $\mathbb{R}^3$ to itself from the points of S and S'. This isometry must send every point of S into a point of S', and every point of $C_S - \{S\}$ into a point of $C_{S'} - \{S'\}$. Moreover, if we construct a vector basis of the convex hull of S (resp. S') from the points of S (resp. S'), then every edge of the convex hull of S must be sent to an edge of the convex hull of S'.

In this way, if the number of points of S and S' is at least five, we have two vector bases with three linearly independent vectors each one, and consequently, we have two bases of $\mathbb{R}^3$. It allows us to construct an isometry from $\mathbb{R}^3$ to itself sending S into S' and C_S into $C_{S'}$. See Figure 3.3 for two examples of this case.

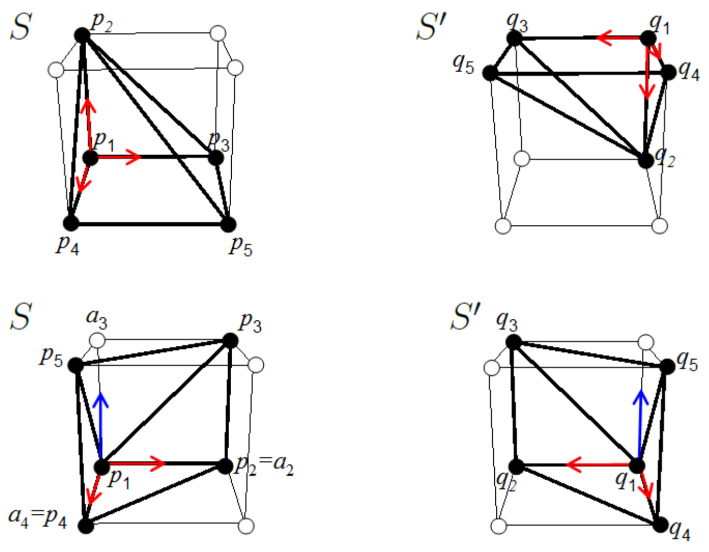


Fig. 3.3: The points of S (resp. S') allows us to construct an isometry from $\mathbb{R}^3$ to itself sending S into S' and C_S into $C_{S'}$.

Otherwise, if the number of points of S and S' is four, the vector basis may only have two linearly independent vectors. In this case, we must choose a fifth vertex $p \in C_S - \{S\}$, in such a way that the isometry must send p into a vertex $q \in C_{S'} - \{S'\}$. This fifth point p (resp. q) allows us to construct, together with the points of S (resp. S'), a basis of $\mathbb{R}^3$ and consequently an isometry from $\mathbb{R}^3$ to itself sending S into S' and C_S into $C_{S'}$. See Figures 3.4 and 3.5 for two examples of this case.

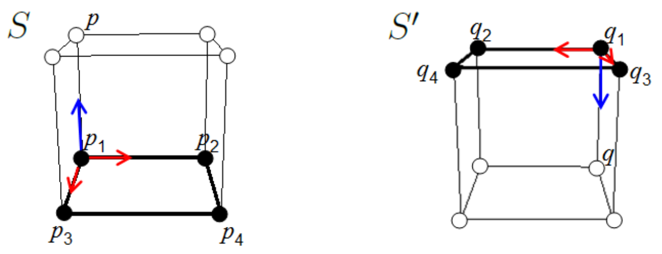


Fig. 3.4: There are two isometries of $\mathbb{R}^3$ to itself sending S into S'. Moreover, by choosing a point $p \in C_S - \{S\}$ (resp. $q \in C_{S'} - \{S'\}$), the points of S (resp. S') together with p (resp. q) allow us to construct an isometry from $\mathbb{R}^3$ to itself sending S into S' and C_S into $C_{S'}$.

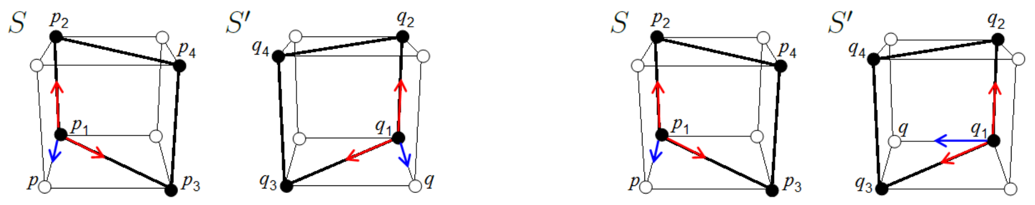


Fig. 3.5: There are two isometries of $\mathbb{R}^3$ to itself sending S into S'. Moreover, both of them send C_S into $C_{S'}$. Consequently, the vertex q is not univocally determined from p, i.e. p can be sent to two different vertices in $C_{S'} - \{S'\}$ by the isometry.

More precisely, the proof of the theorem is as follows.

$\Rightarrow$ We suppose that $S = (p_i)_i$ and $S' = (q_i)_i$ are two isomorphic subsets with $4 \leq c \leq 8$ points each one. Let G_S and $G_{S'}$ be their respective associated multi-graphs. By definition G_S and $G_{S'}$ are isomorphic, so there exists a bijective map $f : (p_i)_i \to (q_i)_i$ preserving the distance between the points of S and S'. Moreover, we can suppose without loss of generality that $f(p_i) = q_i$. It suffices to rearrange the vertices of G_S and $G_{S'}$. In this way $d(p_i, p_j) = d(f(p_i), f(p_j)) = d(q_i, q_j)$.

Starting from $S = \{p_1, ..., p_c\}$ (resp. $S' = \{q_1, ..., q_c\}$) we construct the vector space generated by $\{p_i - p_1\}_{2 \leq i \leq c}$ (resp. $\{q_i - q_1\}_{2 \leq i \leq c}$). A cube has eight vertices and each one of its square faces has four vertices, so

1. If $4 < c \leq 8$, then the vector space generated by $\{p_i - p_1\}_{2 \leq i \leq c}$ (resp. $\{q_i - q_1\}_{2 \leq i \leq c}$) has three linearly independent vectors. In fact, we can assume without loss of generality that these three vectors are determined by the first four points of S (resp. S'). It suffices to rearrange S. Consequently, the vector space $\{p_2 - p_1, p_3 - p_1, p_4 - p_1\}$ (resp. $\{q_2 - q_1, q_3 - q_1, q_4 - q_1\}$) is linearly independent and maximal, i.e. it is a basis of $\mathbb{R}^3$. Therefore f is an isometry from $\mathbb{R}^3$ to itself which sends S into S'.

   Next, we show that f also sends C_S into $C_{S'}$.

   (a) We suppose that p_1, p_2, p_3, p_4 satisfy $d(p_i, p_1) = 1$ for $i = 2, 3, 4$, and let p be a vertex of C_S. We prove that $f(p)$ is a vertex of $C_{S'}$: since $p \in C_S \subset \mathbb{R}^3$ and $\{p_2 - p_1, p_3 - p_1, p_4 - p_1\}$ is a basis of $\mathbb{R}^3$, we can write the vector $p - p_1$ as a finite linear combination of the elements of the basis, i.e. $p - p_1 = \sum_{i=2}^{4} \lambda_i (p_i - p_1)$, with $\lambda_i = 0, 1$ for $i = 2, 3, 4$. By applying f, we obtain $f(p - p_1) = f(\sum_{i=2}^{4} \lambda_i (p_i - p_1)) = \sum_{i=2}^{4} f(\lambda_i (p_i - p_1)) = \sum_{i=2}^{4} \lambda_i f(p_i - p_1) = \sum_{i=2}^{4} \lambda_i (q_i - q_1)$, i.e. the vector $f(p - p_1) = f(p) - f(p_1)$ is a finite linear combination (with the same coefficients $\lambda_i = 0, 1$ for

$i = 2, 3, 4$) of the basis vectors $\{q_2 - q_1, q_3 - q_1, p_4 - q_1\}$, which satisfy $d(q_i, q_1) = d(f(p_i), f(p_1)) = d(p_i, p_1) = 1$ for $i = 2, 3, 4$. Consequently, $f(p)$ is a vertex of $C_{S'}$. See on the top in Figure 3.3 for an example.

(b) Otherwise, we choose a_2, a_3, a_4 such that $d(a_i, p_1) = 1$ for $i = 2, 3, 4$, and we proceed with the points p_1, a_2, a_3, a_4 in a similar way as in (a). Let a_2, a_3, a_4 be the three vertices of C_S satisfying that $d(a_i, p_1) = 1$ for $i = 2, 3, 4$. Since $a_i \in C_S \subset \mathbb{R}^3$ and $\{p_2 - p_1, p_3 - p_1, p_4 - p_1\}$ is a basis of $\mathbb{R}^3$, we can write the vector $a_i - p_1$ as a finite linear combination of the elements of the basis, i.e. $a_i - p_1 = \sum_{i=2}^{4} \alpha_i (p_i - p_1)$ for $i = 2, 3, 4$. Moreover, $\{a_2 - p_1, a_3 - p_1, a_4 - p_1\}$ is a basis of $\mathbb{R}^3$ satisfying the conditions in (a). See on the bottom in Figure 3.3 for an example.

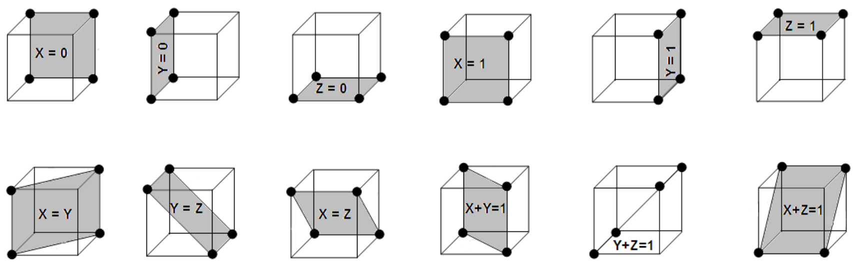
2. If $c = 4$, then $S = \{p_1, p_2, p_3, p_4\}$ (resp. $S' = \{q_1, q_2, q_3, q_4\}$).

   (a) If the three vectors of the vector space generated by $\{p_i - p_1\}_{2 \leq i \leq 4}$ (resp. $\{q_i - q_1\}_{2 \leq i \leq 4}$) are linearly independent, then it will be a basis of $\mathbb{R}^3$. By reasoning as in Case 1, we deduce that f is an isometry from $\mathbb{R}^3$ to itself which sends S into S' and C_S into $C_{S'}$.

   (b) If only two vectors of the vector space generated by $\{p_i - p_1\}_{2 \leq i \leq 4}$ (resp. $\{q_i - q_1\}_{2 \leq i \leq 4}$) are linearly independent, then S (resp. S') is made up by four coplanar points. In a cube, there exist two types of 4–tuple of coplanar vertices, those corresponding to vertices of outer faces, and those corresponding to vertices of inner faces (see Table 3.1). The Euclidean distance between the pairs of vertices of an outer face is 1 and $\sqrt{2}$, whereas the Euclidean distance between the pairs of vertices of an inner face is 1, $\sqrt{2}$ and $\sqrt{3}$. The points of S and S' have the same pairwise distances, so we can suppose that the points of S and S' are the vertices of an outer (resp. inner) face of a cube. In this case, we must choose a fifth vertex $p \in C_S - \{S\}$ and a fifth vertex $q \in C_{S'} - \{S'\}$ which allows us to construct two base of $\mathbb{R}^3$. Moreover, p and q must satisfy that the distance between every point $p_i \in S$ and p coincides with the distance between $q_i \in S'$ and q[1]. This fifth point p (resp. q), together with the points of S (resp. S'), allows us to obtain a third vector linearly independent with the two (linearly independent) vectors constructed from the points of S (resp. S'). Therefore f is an isometry from $\mathbb{R}^3$ to itself which sends S into

1. a constructive proof shows that there always exist two points p and q satisfying these conditions

S'. Moreover, f also sends C_S into $C_{S'}$ since f sends five vertices of C_S (p, p_1, p_2, p_3, p_4) into five vertices of $C_{S'}$ (q, q_1, q_2, q_3, q_4). In this way, we are under the assumptions of Case 1. See Figures 3.4 and 3.5 for examples of this case.



Tab. 3.1: On the top: the six outer faces of the unit cube are determined by the intersection of the unit cube with the planes $X = 0$, $Y = 0$, $Z = 0$, $X = 1$, $Y = 1$, $Z = 1$, respectively. On the bottom: the six inner square faces of the unit cube are determined by the intersection of the unit cube with the planes $X = Y$, $Y = Z$, $X = Z$, $X + Y = 1$, $Y + Z = 1$, $X + Z = 1$, respectively.

$\Leftarrow$ We suppose that there exists an isometry σ which sends S into S' and C_S into $C_{S'}$. Then the points of both S and S' have the same pairwise distances. We construct the multi-graph G_S (resp. $G_{S'}$) associated with S (resp. S'), whose vertices are the points of S (resp. S') and the number of edges between each pair of vertices of G_S (resp. $G_{S'}$) is given by the number of different coordinates between both vertices. Moreover, the number of different coordinates between two vertices of G_S (resp. $G_{S'}$) coincides with the square of the Euclidean distance between them. In this way, the isometry σ preserves the number of edges between each pair of vertices of G_S and $G_{S'}$. Consequently, these two multi-graphs are isomorphic to each other. Hence, S and S' are isomorphic.

$\square$

Remark 3.1.4. *Let us note that any isometry of $\mathbb{R}^3$ to itself which sends a subset S of a cube C_S into a subset S' of a cube $C_{S'}$, it also sends C_S into $C_{S'}$, regardless of the number of points of the subsets. In this way, Theorem 3.1.2 is true for subsets of a cube with $0 \leq c \leq 8$ points. However, in Chapter 4, we show that a generalization to higher dimension of Theorem 3.1.2 is not true if we remove the hypothesis of cardinality. More concretely, we show that there exists an isometry of $\mathbb{R}^4$ to itself which sends a subset S made up by 4 vertices of a 4–cube HC_S into a subset S' made up by 4 vertices of a 4–cube $HC_{S'}$, and it does not send HC_S into $HC_{S'}$.*

Taking into account the previous results, we arrange the set V of vertices of the unit cube on an order relation $\prec$ (for instance, the lexicographic order) and we conceive an algorithm which uses as input the ordered set $(V, \prec)$. For $4 \leq c \leq 8$, this algorithm: (a) constructs an ordered set $(V_c, \prec)$ containing the $C(8, c)$ subsets with c points; (b) associates each subset $(V_c)_i \subset V_c$ with a multi-graph $(G_{V_c})_i$. The vertices of $(G_{V_c})_i$ are the points of $(V_c)_i$, and the number of edges between each pair of vertices $p_k, p_l \in (G_{V_c})_i$ coincides with the number of different coordinates between both vertices; and (c) checks if there exists an isomorphism between each pair $(G_{V_c})_{i_1}, (G_{V_c})_{i_2}$ of multi-graphs associated with two subsets $(V_c)_{i_1}, (V_c)_{i_2} \subset V_c$ which satisfy $(V_c)_{i_1} \prec (V_c)_{i_2}$. In Example 3.1.5 the procedure for $c = 7$ is described with detail.

Example 3.1.5. *Let* $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\} = \{\{0, 0, 0\}, \{0, 0, 1\}, \{0, 1, 0\}, \{0, 1, 1\}, \{1, 0, 0\}, \{1, 0, 1\}, \{1, 1, 0\}, \{1, 1, 1\}\}$ *be the set of vertices of the unit cube arranged on lexicographic order* ($\prec_{lex}$).

First, we consider the ordered set $(V_7, \prec_{lex})$ *containing the eight subsets with seven points, i.e.* $V_7 = \{\{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}, \{v_1, v_2, v_3, v_4, v_5, v_6, v_8\}, \{v_1, v_2, v_3, v_4, v_5, v_7, v_8\}, \{v_1, v_2, v_3, v_4, v_6, v_7, v_8\}, \{v_1, v_2, v_3, v_5, v_6, v_7, v_8\}, \{v_1, v_2, v_4, v_5, v_6, v_7, v_8\}, \{v_1, v_3, v_4, v_5, v_6, v_7, v_8\}, \{v_2, v_3, v_4, v_5, v_6, v_7, v_8\}\}$.

Next, for $i = 1, ..., 8$ *each subset* $(V_7)_i \subset V_7$ *is associated with a multi-graph* $(G_{V_7})_i$. *The vertices of* $(G_{V_7})_i$ *are the points of* $(V_7)_i$ *and the number of edges between two vertices* $v_k, v_l \in (G_{V_7})_i$ *is determined by the number of different coordinates between* v_k *and* v_l. *These multi-graphs are shown in Figure 3.6.*

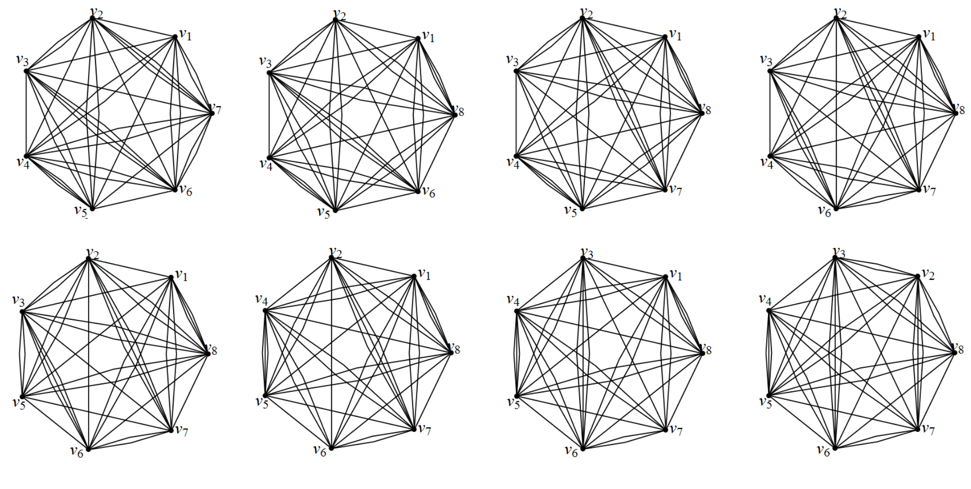
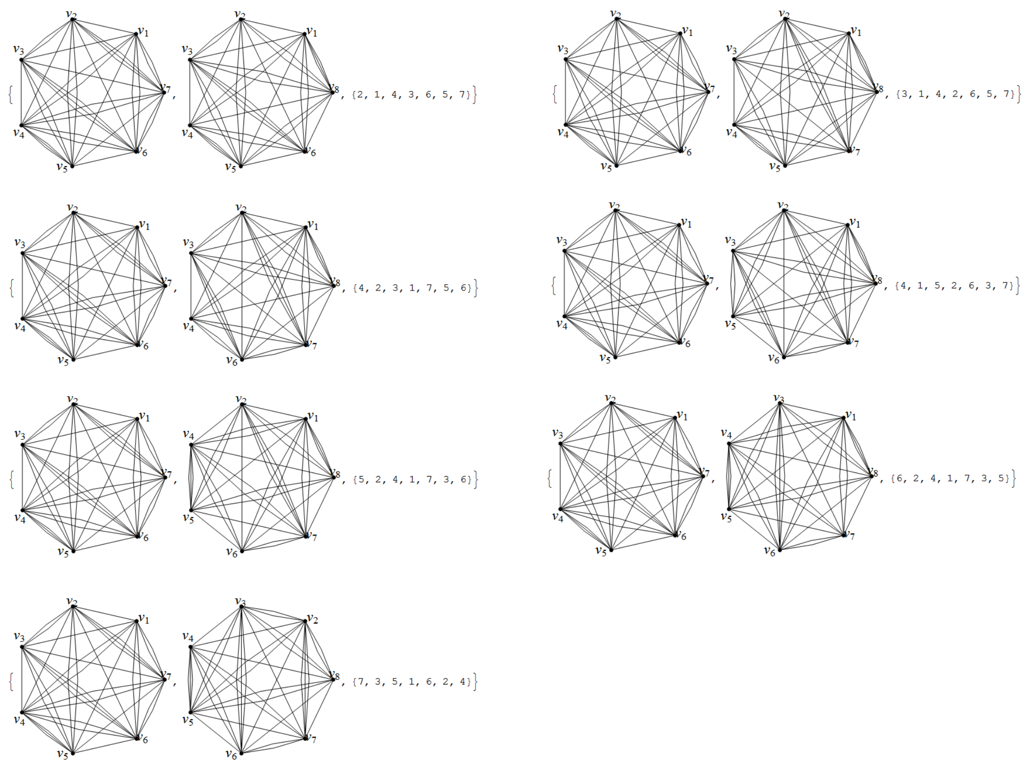


Fig. 3.6: Multi-graphs associated with the subsets of V_7.

Finally, it is checked if there exists an isomorphism between each pair $(G_{V_7})_{i_1}, (G_{V_7})_{i_2}$ of multi-graphs associated with two subsets $(V_7)_{i_1}, (V_7)_{i_2} \subset V_7$ which satisfy $(V_7)_{i_1} \prec_{lex} (V_7)_{i_2}$. In Table 3.2, the isomorphisms between some pairs of these multi-graphs are shown.



Tab. 3.2: Given $(G_{V_7})_1$ and $(G_{V_7})_i$, the third element of each list shows a rearrangement of the vertices of the multi-graph $(G_{V_7})_1$ for obtaining the multi-graph $(G_{V_7})_i$, with $i = 2, ..., 8$. Algorithm 3.1.1 shows that there exists only one pattern subset with seven points.

Algorithm 3.1.1 determines the pattern subsets with $4 \leq c \leq 8$ points. The pattern subsets with $0 \leq c < 4$ points are determined by complementation. In Section 3.1.2, we will associate a *pattern cell* with each pattern subset.

Algorithm 3.1.1

Input: $(V, \prec)$ set of vertices of the unit cube together with an order relation $\prec$.
Output: pattern subsets with $4 \leq c \leq 8$ points.
begin
// PS: empty list to store the vertices of the non-isomorphic multi-graphs.

```
1: for $c = 4, ..., 8$ do
```

```
2:     Construct an ordered set $(V_c, \prec)$ containing the $C(8, c)$ subsets with $c$ points
3:     for each $(V_c)_i \subset V_c$ do
4:        Determine the multi-graph $(G_{V_c})_i$ associated with $(V_c)_i$
5:     end for
6:     for all $(V_c)_{i_1} \subset V_c$ and $(V_c)_{i_2} \subset V_c$ such that $(V_c)_{i_1} \prec (V_c)_{i_2}$ do
7:        if $(G_{V_c})_{i_1}$ and $(G_{V_c})_{i_2}$ are isomorphic then $\{(V_c)_{i_1}$ and $(V_c)_{i_2}$ are isometric.$\}$
8:           $V_c = V_c - \{(V_c)_{i_2}\}$
9:        end if
10:    end for
11:    $PS = PS \bigcup V_c$
12: end for
13: return $PS$
end
```

Remark 3.1.6. *Given an order relation $\prec$ on the set of vertices of the unit cube, Algorithm 3.1.1 determines the smallest pattern subsets with respect to $\prec$. Moreover, by changing the order relation, other pattern subsets (isometric to those shown in Table 3.3) are obtained.*

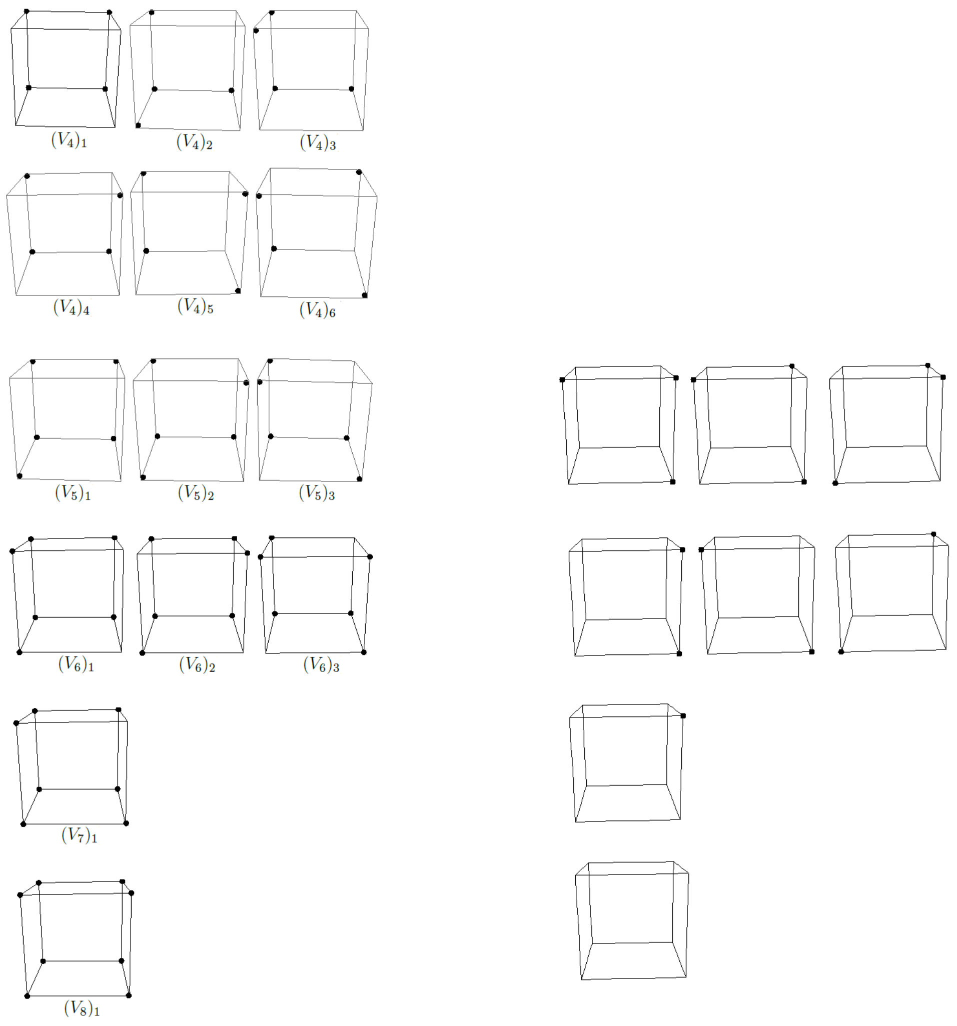
The results shown in Theorem 3.1.7 are determined by Algorithm 3.1.1.

Theorem 3.1.7. *In $\mathbb{Z}^3$, there exist: (a) six pattern subsets with four points; (b) three pattern subsets with five points; (c) three pattern subsets with six points; (d) one pattern subset with seven points; and (e) one pattern subset with eight points.*

Taking into account the complementation, we can formulate Corollary 3.1.8.

Corollary 3.1.8. *In $\mathbb{Z}^3$, there exist: (b') three pattern subsets with three points; (c') three pattern subsets with two points; (d') one pattern subset with one point; and (e') one pattern subset with zero points.*

Remark 3.1.9. *Let us note that the results shown in Theorem 3.1.7 and Corollary 3.1.8 coincide with those obtained by using marching cubes algorithm and Kenmochi et al. method except for the subsets with four points.*



Tab. 3.3: On the left: pattern subsets with $4 \leq c \leq 8$ points, obtained by using lexicographic order in Algorithm 3.1.1. On the right: pattern subsets with $0 \leq c < 4$ points, obtained by complementation.

3.1.1.2 A second solution

Algorithm 3.1.1 is based on graph isomorphisms, which is a problem in NP (see [3, 20] for more details). For this reason, a more efficient algorithm for finding the pattern subsets has been implemented. The algorithm defined in this paragraph computes the isometric subsets in a direct way. More concretely, the group of isometries of a cube is determined and applied to the subsets of points. In this way, we obtain the subsets isometric to a given subset, and consequently, the pattern subsets. A scheme of this algorithm is shown in Figure 3.7.

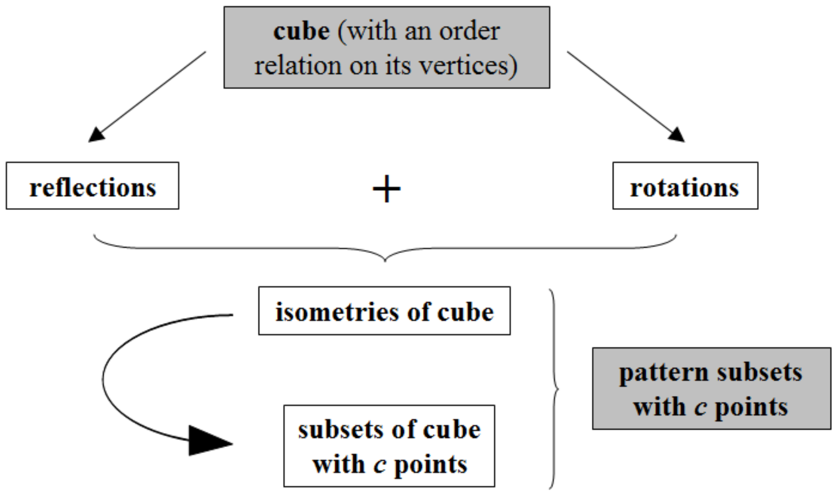

cube (with an order relation on its vertices)
reflections
+
rotations
isometries of cube
subsets of cube with c points
pattern subsets with c points

Fig. 3.7: Scheme of a more efficient algorithm than Algorithm 3.1.1 for finding the pattern subsets.

Computing the group of isometries of a cube

The group *iso_cube* of isometries of a cube consists of rigid motions (rotations, reflections and translations) leaving it invariant. Moreover, there do not exist translations leaving a cube invariant, so *iso_cube* is only made up by the rotations and reflections which leave a cube invariant.

Rotations are determined by angles and rotation axes. The points of the axis are fixed under the rotation. If the rotation leaves a cube invariant, then the axis must go through the cube passing by its central point. Below, we show some well-known results (see [2, 13]) about the angle and the axis of each of the rotations which leave a cube invariant.

– Rotation about an axis from the center of a face to the center of the opposite face by an angle of $\pi/2$, $-\pi/2$ or π radians: there exist 3 axes (see Figure 3.8), so there are 3 rotations of this type for each angle, altogether 9 rotations of this type.

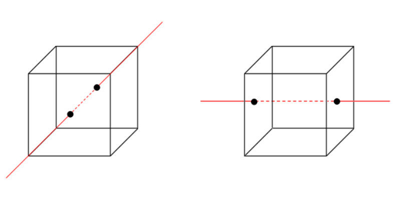


Fig. 3.8: The rotation determined by each of the red axes and by an angle of $\pi/2$, $-\pi/2$ and π radians, respectively, leaves the cube invariant.

– Rotation about an axis from the center of an edge to the center of the opposite edge by an angle of π radians: there exist 6 axes (see Figure 3.9) and 1 rotation per axis, so there are 6 rotations of this type.

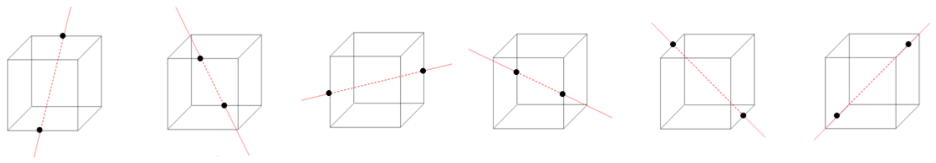


Fig. 3.9: The rotation determined by each of the red axes and by an angle of π radians leaves the cube invariant.

– Rotation about an axis from one vertex to the opposite vertex by an angle of $\frac{2}{3}\pi$ or $-\frac{2}{3}\pi$ radians: there exist 4 axes (see Figure 3.10), so there are 4 rotations of this type for each angle, altogether 8 rotations of this type.

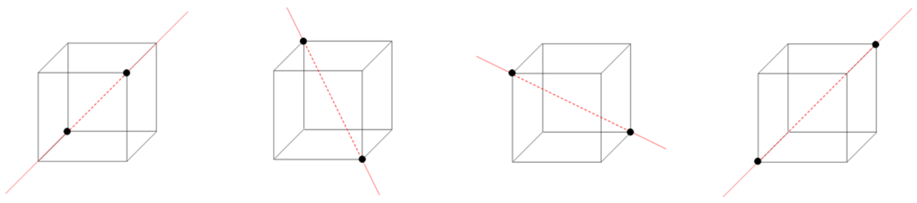


Fig. 3.10: The rotation determined by each of the red axes and by an angle of $\frac{2}{3}\pi$ radians (resp. $-\frac{2}{3}\pi$ radians) leaves the cube invariant.

Reflections are determined by reflection planes. The points of this plane are fixed under the reflection. If the reflection leaves a cube invariant, then the plane must contain the central point of the cube and to split it into two identical polyhedra. Below, we show some well-known results (see [2, 13]) about the reflection plane of each of the reflections which leave a cube invariant.

– 3 reflections with planes midway between pairs of opposite faces (see Figure 3.11).

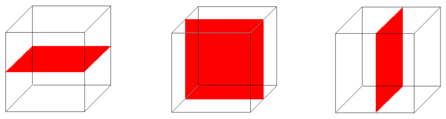


Fig. 3.11: The reflection determined by each of the red planes leaves the cube invariant.

– 6 reflections with planes passing through pairs of opposite edges (see Figure 3.12).

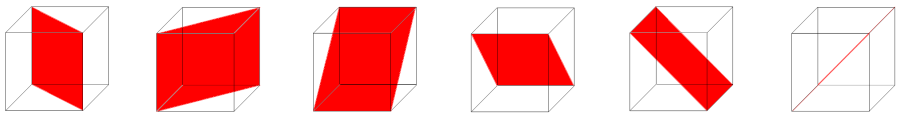


Fig. 3.12: The reflection determined by each of the red planes leaves the cube invariant.

Let us note that the group of the isometries of a cube is made up by 48 elements, namely: (a) the identity map; (b) 23 rotations (see Figures 3.8, 3.9 and 3.10); (c) 9 reflections (see Figures 3.11 and 3.12); and (d) 15 compositions of one rotation with one reflection such that the rotation axis and the reflection plane are incident at a point. See [13] for more details.

Remark 3.1.10. *Let $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$ be the set of vertices of the unit cube.* ***Automorphisms[Hypercube[3]]*** *in the* Mathematica *package* ***Combinatorica'*** *gives the following list with the 48 permutations of vertices which leave the unit cube invariant.*
$\{\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}, \{v_1, v_2, v_7, v_8, v_5, v_6, v_3, v_4\}, \{v_1, v_3, v_2, v_4, v_8, v_6, v_7, v_5\},$
$\{v_1, v_3, v_7, v_5, v_8, v_6, v_2, v_4\}, \{v_1, v_7, v_2, v_8, v_4, v_6, v_3, v_5\}, \{v_1, v_7, v_3, v_5, v_4, v_6, v_2, v_8\},$
$\{v_2, v_1, v_4, v_3, v_6, v_5, v_8, v_7\}, \{v_2, v_1, v_8, v_7, v_6, v_5, v_4, v_3\}, \{v_2, v_4, v_1, v_3, v_7, v_5, v_8, v_6\},$
$\{v_2, v_4, v_8, v_6, v_7, v_5, v_1, v_3\}, \{v_2, v_8, v_1, v_7, v_3, v_5, v_4, v_6\}, \{v_2, v_8, v_4, v_6, v_3, v_5, v_1, v_7\},$
$\{v_3, v_1, v_4, v_2, v_6, v_8, v_5, v_7\}, \{v_3, v_1, v_5, v_7, v_6, v_8, v_4, v_2\}, \{v_3, v_4, v_1, v_2, v_7, v_8, v_5, v_6\},$
$\{v_3, v_4, v_5, v_6, v_7, v_8, v_1, v_2\}, \{v_3, v_5, v_1, v_7, v_2, v_8, v_4, v_6\}, \{v_3, v_5, v_4, v_6, v_2, v_8, v_1, v_7\},$

$\{v_4, v_2, v_3, v_1, v_5, v_7, v_6, v_8\}, \{v_4, v_2, v_6, v_8, v_5, v_7, v_3, v_1\}, \{v_4, v_3, v_2, v_1, v_8, v_7, v_6, v_5\},$
$\{v_4, v_3, v_6, v_5, v_8, v_7, v_2, v_1\}, \{v_4, v_6, v_2, v_8, v_1, v_7, v_3, v_5\}, \{v_4, v_6, v_3, v_5, v_1, v_7, v_2, v_8\},$
$\{v_5, v_3, v_6, v_4, v_8, v_2, v_7, v_1\}, \{v_5, v_3, v_7, v_1, v_8, v_2, v_6, v_4\}, \{v_5, v_6, v_3, v_4, v_1, v_2, v_7, v_8\},$
$\{v_5, v_6, v_7, v_8, v_1, v_2, v_3, v_4\}, \{v_5, v_7, v_3, v_1, v_4, v_2, v_6, v_8\}, \{v_5, v_7, v_6, v_8, v_4, v_2, v_3, v_1\},$
$\{v_6, v_4, v_5, v_3, v_7, v_1, v_8, v_2\}, \{v_6, v_4, v_8, v_2, v_7, v_1, v_5, v_3\}, \{v_6, v_5, v_4, v_3, v_2, v_1, v_8, v_7\},$
$\{v_6, v_5, v_8, v_7, v_2, v_1, v_4, v_3\}, \{v_6, v_8, v_4, v_2, v_3, v_1, v_5, v_7\}, \{v_6, v_8, v_5, v_7, v_3, v_1, v_4, v_2\},$
$\{v_7, v_1, v_5, v_3, v_6, v_4, v_8, v_2\}, \{v_7, v_1, v_8, v_2, v_6, v_4, v_5, v_3\}, \{v_7, v_5, v_1, v_3, v_2, v_4, v_8, v_6\},$
$\{v_7, v_5, v_8, v_6, v_2, v_4, v_1, v_3\}, \{v_7, v_8, v_1, v_2, v_3, v_4, v_5, v_6\}, \{v_7, v_8, v_5, v_6, v_3, v_4, v_1, v_2\},$
$\{v_8, v_2, v_6, v_4, v_5, v_3, v_7, v_1\}, \{v_8, v_2, v_7, v_1, v_5, v_3, v_6, v_4\}, \{v_8, v_6, v_2, v_4, v_1, v_3, v_7, v_5\},$
$\{v_8, v_6, v_7, v_5, v_1, v_3, v_2, v_4\}, \{v_8, v_7, v_2, v_1, v_4, v_3, v_6, v_5\}, \{v_8, v_7, v_6, v_5, v_4, v_3, v_2, v_1\}\}$

Let us note that each of these permutations corresponds to an isometry of the unit cube.

Remark 3.1.11. *A standard implementation for computing the permutations of vertices which leave the unit cube C invariant can be conceived as follows.*

Let $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$ be the set of vertices of C: (a) v_1 can be placed on eight different positions; (b) if v_2 is the end-point of an edge incident to v_1, then v_2 can be placed on three different positions since there exist three edges incident to v_1; (c) if v_3 is the end-point of another edge incident to v_1, then (once v_2 is placed) v_3 can be placed on two different positions; (d) if v_4 is the end-point of another edge incident to v_1, then (once v_2, v_3 are placed) v_4 can be placed on only one position; analogously, (e) the positions of the vertices v_5, v_6, v_7, v_8 are fixed.

In this way, we can conclude that there exist $8 \cdot 3 \cdot 2 \cdot 1 = 48$ elements in the group of isometries of C.

Applying the group of isometries to the subsets

Once computed the group of isometries *iso_cube* of a cube, we define Algorithm 3.1.2 for obtaining the pattern subsets from this group.

Algorithm 3.1.2

```
Input: $(V, \prec)$ set of vertices of the unit cube together with an order relation $\prec$.
       iso_cube: group of isometries of the unit cube.
Output: pattern subsets with $0 \leq c \leq 8$ points.
begin
// PS: empty list to store the pattern subsets with $0 \leq c \leq 8$ points.
1: for $c = 0, ..., 8$ do
2:     Construct an ordered set $(V_c, \prec)$ containing the $C(8, c)$ subsets with $c$ points
3:     for each $(V_c)_i \subset V_c$ ordered by $\prec$ do
4:         for each $\sigma \in$ iso_cube $- \{identity\}$ do
```

```
5:          $V_c = V_c - \{\sigma((V_c)_i)\}$
6:        end for
7:    end for
8:    $PS = PS \bigcup V_c$
9: end for
10: return $PS$
end
```

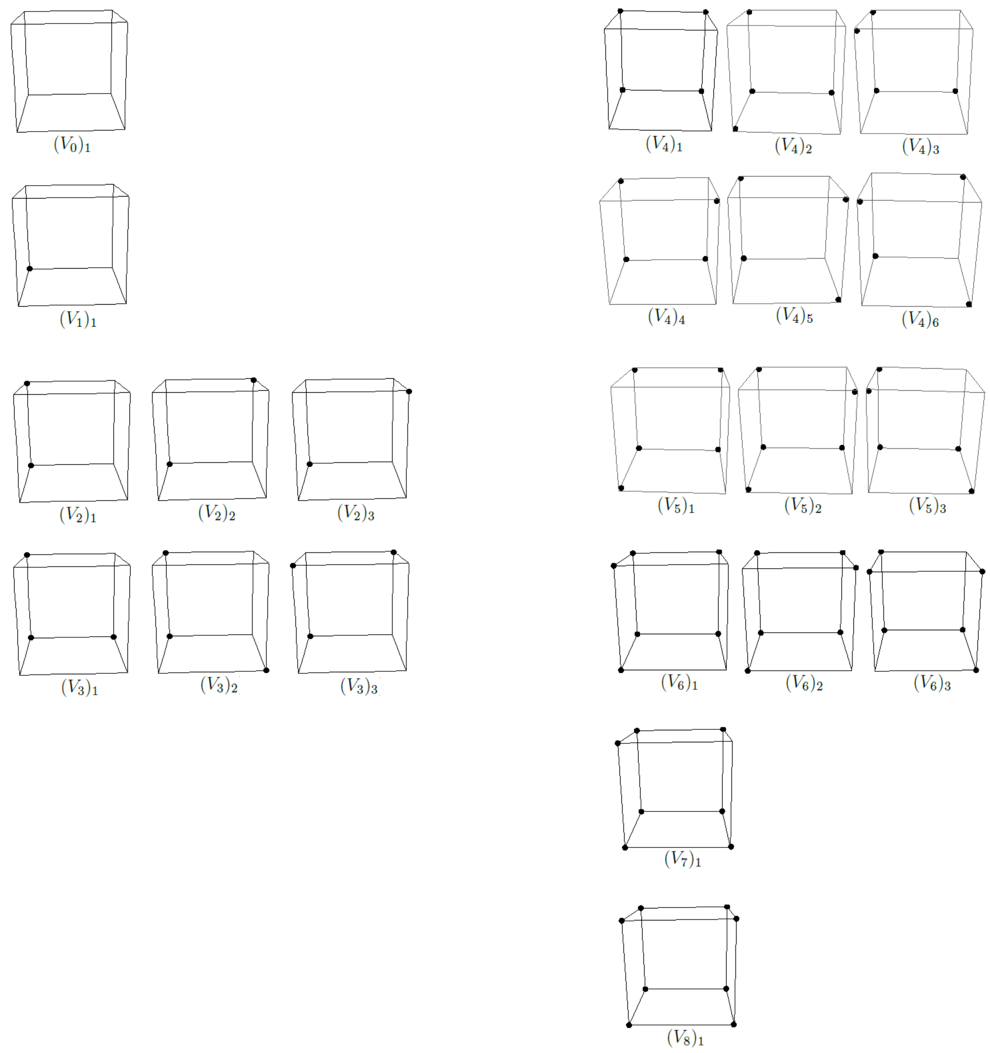
First, Algorithm 3.1.2 constructs an ordered set $(V_c, \prec)$ containing the $C(8, c)$ subsets with $0 \leq c \leq 8$ vertices of the unit cube. Next, for $(V_c)_i \subset V_c$, it computes and removes any subset $(V_c)_j \neq (V_c)_i$ isometric to $(V_c)_i$. In this way, Algorithm 3.1.2 allows us to obtain a representative subset of each isometry class.

Remark 3.1.12. *Algorithm 3.1.2 improves the computational time of Algorithm 3.1.1 (see Table 3.4).*

	Algorithm 3.1.1	**Algorithm 3.1.2**
$c = 4$	3.12 sec.	0.31 sec.
$c = 5$	3.83 sec.	0.29 sec.
$c = 6$	2.8 sec.	0.26 sec.
$c = 7$	1.58 sec.	0.21 sec.

Tab. 3.4: Algorithm 3.1.1 constructs and checks $C(8, 4) = 70$, $C(8, 5) = 56$, $C(8, 6) = 28$ and $C(8, 7) = 8$ multi-graphs in 3.12, 3.83, 2.8 and 1.58 seconds of CPU time, respectively. These results are improved by Algorithm 3.1.2 to 0.31, 0.29, 0.26 and 0.21 seconds of CPU time, respectively. The computations have been made by using Mathematica 7.0.0 in AMD Turion(tm) 64 X2 Mobile Technology TL-58 1.90 GHz.

Remark 3.1.13. *In the same way as Algorithm 3.1.1, Algorithm 3.1.2 determines the smallest pattern subsets with respect to the order relation given on the set of vertices of the unit cube; so that, by changing the order relation, other pattern subsets (isometric to those shown in Table 3.5) are obtained.*



Tab. 3.5: Pattern subsets in $\mathbb{Z}^3$ obtained by using the lexicographic order in Algorithm 3.1.2. The pattern subsets on the left coincide (up to isometries) with those on the right in Table 3.3. The pattern subsets on the right coincide with those on the left in Table 3.3.

Remark 3.1.14. *Theorem 3.1.7 and Corollary 3.1.8 can be proved by using Algorithm 3.1.2.*

Remark 3.1.15. *Let us observe that the twenty-two pattern subsets shown in Table 3.5 coincide (up to isometries) with the twenty-two unit cells presented by Kong and Roscoe in [17] (see Table 2.9). More concretely, the pattern subset* $(V_0)_1$, $(V_1)_1$, $(V_2)_1$, $(V_2)_2$, $(V_2)_3$, $(V_3)_1$, $(V_3)_2$, $(V_3)_3$, $(V_4)_1$, $(V_4)_2$, $(V_4)_3$,

$(V_4)_4$, $(V_4)_5$, $(V_4)_6$, $(V_5)_1$, $(V_5)_2$, $(V_5)_3$, $(V_6)_1$, $(V_6)_2$, $(V_6)_3$, $(V_7)_1$, $(V_8)_1$, *coincides with the unit cell labeled by 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 17, 15, 16, 20, 19, 18, 21, 22, respectively, in Table 2.9.*

Below, we associate each of the 22 pattern subsets with a *pattern cell.* Every pattern cell consists in the convex hull of the points of the corresponding pattern subset. More precisely, every pattern subset is associated with (1) an open cell made up by the points inside of the convex hull, and (2) a boundary made up by the boundary cells of the convex hull.

3.1.2 Pattern cells

We show a computational method for determining the convex hull of each of the pattern subsets. The technique consists in deforming the cube which contains the subset. Moreover, we deform the faces of the cube which contain white vertices (see Figure 3.13 (a)). These deformations are a consequence of degenerating edges incident to white vertices. These edge degeneracies can lead to the fact that a face (resp. volume) degenerates into an edge (resp. face) incident to it (see Figure 3.13 (b) and (c)).

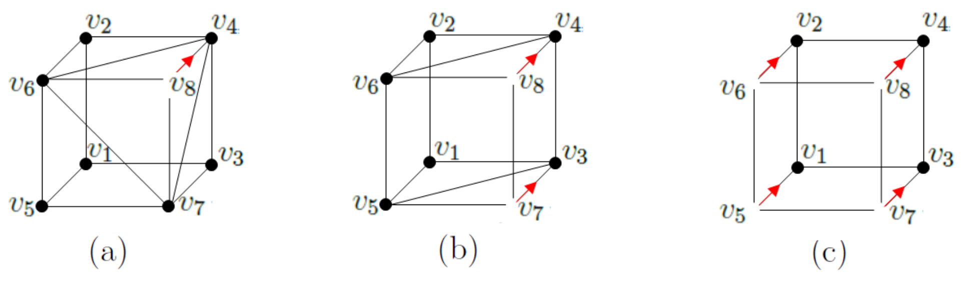


Fig. 3.13: (a) Degeneracy of the edge v_4v_8 into the vertex v_4. This edge degeneracy leads to the fact that the square faces $v_2v_4v_6v_8$ and $v_3v_4v_7v_8$ are deformed into the triangular faces $v_2v_4v_6$ and $v_3v_4v_7$, respectively. Moreover, the square face $v_5v_6v_7v_8$ is geometrically deformed into a fold face; it leads to the fact that this face is subdivided into two triangular faces $v_4v_6v_7$ and $v_5v_6v_7$. (b) Degeneracies of the edges v_3v_7 and v_4v_8 into the vertices v_3 and v_4, respectively. These edge degeneracies lead to the fact that the square face $v_3v_4v_7v_8$ degenerates into the edge v_3v_4. (c) Degeneracies of the edges v_1v_5, v_2v_6, v_3v_7 and v_4v_8 into the vertices v_1, v_2, v_3 and v_4, respectively. These edge degeneracies lead to the fact that the cube degenerates into the square face $v_1v_2v_3v_4$.

3.1.2.1 Finding the edges to degenerate

We show that by degenerating edges incident to a white vertex which do not belong to the cube, we do not always obtain the convex hull of the pattern subset. Moreover, we justify that if during the deformation of the cube we only degenerate edges belonging to the cube, then we always obtain the convex hull of the subset.

Degenerating edges non-belonging to the cube

We suppose that we want to compute the convex hull of the subset $\{v_1, v_2, v_3, v_4, v_8\}$ of vertices of the cube (see Figure 3.14 (a)), and we suppose that at the penultimate step of the deformation we have obtained the polyhedron shown in Figure 3.14 (b). The last step of the procedure consists in degenerating one of the edges incident to v_5.

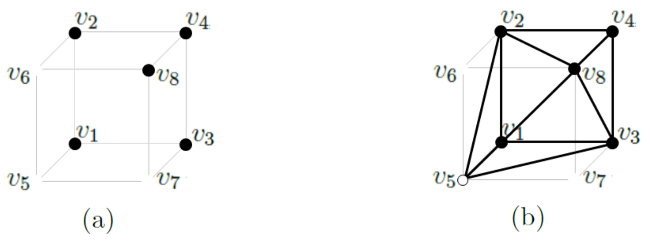
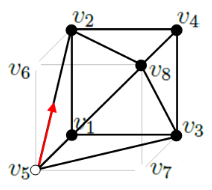
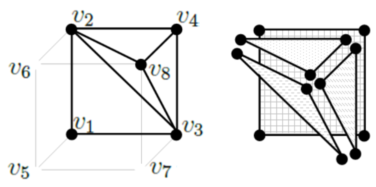


Fig. 3.14: (a) Subset $\{v_1, v_2, v_3, v_4, v_8\}$ of vertices of the cube. (b) Result obtained at the penultimate step of the deformation.

If we degenerate the edge v_2v_5 (see Figure 3.15 (a)), we do not obtain the convex hull of the subset of black points. Actually, the black points define a volume contained in the cube. However, in Figure 3.15 (b) we have obtained an object made up by the square face $v_1v_2v_3v_4$ (containing the triangular face $v_1v_2v_3$) and three non-coplanar triangular faces $v_2v_3v_8$, $v_2v_4v_8$ and $v_3v_4v_8$ attached each other by an edge, containing the volume incident to these three faces and a half of the square. It is because to degenerate the edge v_2v_5, the edge v_3v_5 goes through the polyhedron shown in Figure 3.14 (b), sending it to the volume $v_2v_3v_4v_8$ together with a face $v_1v_2v_3$ out of the volume.



(a)



(b)

Fig. 3.15: (a) By degenerating the edge v_2v_5, we do not obtain the convex hull of the points in Figure 3.14 (a). (b) We obtain an object made up by a square face $v_1v_2v_3v_4$ and three non-coplanar triangular faces $v_2v_3v_8$, $v_2v_4v_8$ and $v_3v_4v_8$, containing the volume incident to these three faces and a half of the square.

Let us note that we obtain a similar result by degenerating the edge v_3v_5. On the other hand, if we degenerate the edge v_1v_5 or v_5v_8, we obtain the convex hull of the black points (see Figure 3.16).

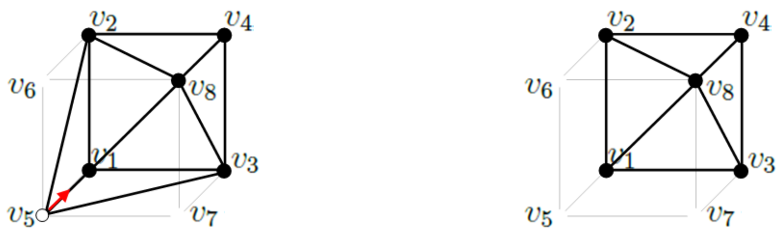


Fig. 3.16: By degenerating the edge v_1v_5, we obtain the convex hull of the points in Figure 3.14 (a).

Degenerating edges belonging to the cube

We show that by degenerating an edge belonging to the cube, at each step, we obtain the convex hull. Moreover, taking into account that a cube can be decomposed into two prisms with triangular bases (see Figure 3.17), by symmetries, it suffices to prove that by degenerating an edge of these prisms, at each step, we obtain the convex hull.

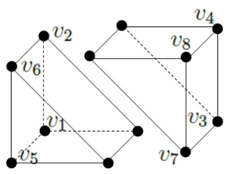


Fig. 3.17: A cube decomposed into two prisms with triangular bases.

At the first step of the deformation of a cube C, we consider the prism D containing the white vertex B. If we degenerate an edge $e \in C \bigcap D$ incident to B, we obtain a pyramid (first two pairs of pictures in Figure 3.18) or into a pyramid with a non-planar convex foursquare face (last four pairs of pictures in Figure 3.18). Latter appears by deforming the square face of D which contains B and it is not incident to e. We can interpret this deformation as a fold face, so it is necessary to introduce an edge for representing the fold. In this way, the non-planar face is replaced by two triangular faces.

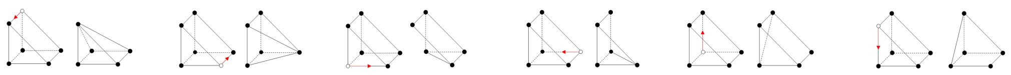


Fig. 3.18: By degenerating any edge of $C \bigcap D$, the prism is deformed into a pyramid.

We suppose, without loss of generality, that at the following step of the deformation, B' is a white vertex of one of the pyramids P obtained in Figure 3.18 by deforming D. Let us note that B' can be the apex or a base vertex of the pyramid P. Below, we show that, in both cases, if we degenerate an edge $e' \in C \bigcap P$, then we obtain the convex hull of the vertices of $P - \{B'\}$.

- If B' is the apex of P, we degenerate an edge $e' \in C \bigcap P$ incident to the apex. Consequently, P degenerates into its square base, and we obtain a square face divided into two coplanar triangular faces sharing an edge. In order to obtain the square face, we remove the edge shared by the two triangular faces (see Figure 3.19).

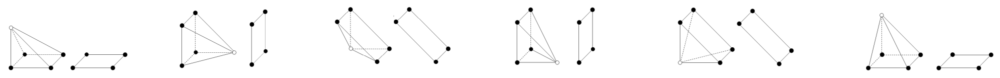


Fig. 3.19: By degenerating any edge of $C \bigcap P$ incident to the apex, the pyramid is degenerated into a square.

- If B' is a vertex of the base of P, we degenerate any edge $e' \in C \bigcap P$ incident to B'. Consequently, P is deformed into a tetrahedron (see first four pairs of pictures in Figure 3.20) or into a tetrahedron with a non-planar convex foursquare face (see last pair of pictures in Figure 3.20). Likewise to the case of the triangular prism, the remaining edge is because it appears a non-planar convex foursquare face. This fact involves to have to introduce the edge which represents the fold of the face.

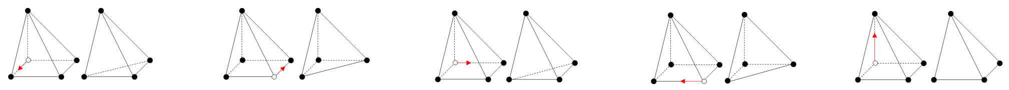


Fig. 3.20: By degenerating any edge of C incident to two vertices of the base of P, we obtain a tetrahedron. By degenerating any edge of C which joins a vertex of the base of P with the apex, we obtain a tetrahedron (except for an edge).

Finally, the degeneracy of a tetrahedron into a triangle, a triangle into an edge, and an edge into a point, respectively, is obtained trivially by degenerating an edge belonging to the cube incident to a white vertex (see Figure 3.21).

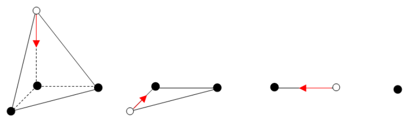


Fig. 3.21: From left to right: degeneracy of a tetrahedron into a triangle, a triangle into an edge, and an edge into a point.

Remark 3.1.16. *If, at each step, we degenerate an edge belonging to the cube incident to a white vertex, the previous results prove that we always obtain the convex hull.*

Remark 3.1.17. *It is also possible to obtain the convex hull by degenerating edges which do not belong to the cube; it happened in Figure 3.15 with the edge v_5v_8. In this sense:*

– *Figure 3.18 allows us to note that by degenerating any edge (belonging or not to the cube) of a prism, we obtain a pyramid.*
– *Figure 3.19 shows that if the apex is the white vertex, then the pyramid is degenerated into its basis by degenerating any edge (belonging or not to the cube) of the pyramid incident to the apex.*
– *If the white vertex is a vertex of the base of the pyramid, then by degenerating any edge (belonging or not to the cube) incident to it and to*

another base vertex, we obtain a tetrahedron. It is the case of the edge v_5v_8 *in Figure 3.15, which the vertices* v_1, v_2, v_4, v_5, v_8 *define a pyramid in.*
Moreover, the only operation which does not allow us to obtain a tetrahedron consists in degenerating an edge of the pyramid which does not belong to the cube and it joins the apex with a white vertex of the base. It is the case of the edge v_2v_5 *(resp.* v_3v_5*) in Figure 3.15, which the vertices* v_1, v_2, v_4, v_5, v_8 *(resp.* v_1, v_3, v_4, v_5, v_8*) define a pyramid in. On the other hand, if* v_1 *were the white vertex in Figure 3.14 (b), then by degenerating the edge* v_1v_2 *(resp.* v_1v_3*) we would obtain the convex hull (see Figure 3.22).*

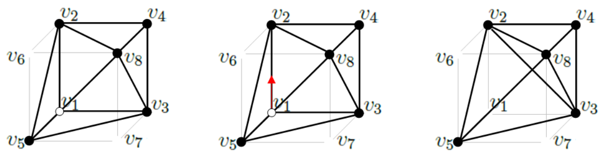


Fig. 3.22: By degenerating the edge v_1v_2, we obtain the convex hull of v_2, v_3, v_4, v_5, v_8.

In the following paragraph, we detail a procedure to assure that at each step of the deformation of the cube, there exists an edge of the cube incident to the white vertex to treat. This edge is degenerated in order to obtain the convex hull.

3.1.2.2 An order relation on the edge degeneracies

We show that the edges of the cube have to degenerate following a certain order; otherwise, in the procedure of deformation of the cube there can be white vertices non-incident to edges of it (see Figure 3.23), in which case it may not be obtained the convex hull of the points of the pattern subset. Moreover, we describe a procedure for obtaining an order relation for degenerating the edges of the cube. This order relation guarantees that, at each step of the deformation of the cube, the white vertex to treat is incident to an edge of the cube, avoiding situations as that shown in Figure 3.23.

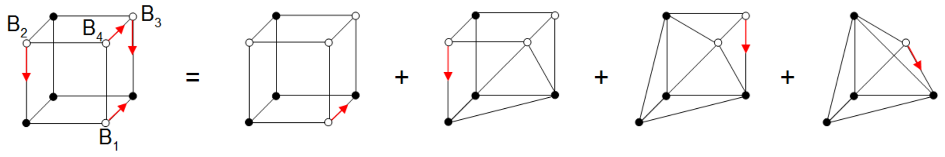


Fig. 3.23: The first picture shows an order for degenerating the edges incident to the white vertices of the cube. The degenerated edges in the second, third, and fourth picture, respectively, lead to the deformation of the three edges of the cube incident to the vertex B_4. There are not any edge of the cube incident to the vertex B_4 for degenerating.

The procedure for obtaining an order relation on the edge degeneracies consists of two steps: (1) considering the white vertices of the cube and the edges which join them as a subgraph S of the cube; and (2) constructing a rooted spanning tree of each connected component of S. These trees allow us to establish a hierarchy on the set of white vertices (the last vertex is the root), and consequently, an order for degenerating the edges which join these vertices. The idea to avoid situations as that shown in Figure 3.23 is to choose as root of each tree a white vertex adjacent to a black vertex of the cube. In this way, each root is always incident to an edge of the cube which has not been deformed.

Let $(V_c)_i \subset \mathbb{Z}^3$ be a pattern subset with $0 \leq c \leq 8$ points, contained in a cube $C_{(V_c)_i}$. First, we consider the cube $C_{(V_c)_i}$ as a graph with eight vertices and twelve edges. Then, we delete the points of $(V_c)_i$ from $C_{(V_c)_i}$ and, consequently, the edges incident to these points. In this way, we obtain a subgraph S of $C_{(V_c)_i}$ with $8-c$ vertices and the remaining edges of $C_{(V_c)_i}$ (see Figure 3.24 for an example). Next, we compute the connected components of the subgraph S. Additionally, for each connected component of S, we construct a rooted spanning tree whose root is a white vertex adjacent to a black vertex of the cube. Let us note that these trees establish a hierarchy on the set of white vertices of $C_{(V_c)_i}$. Moreover, this hierarchy determines an order for degenerating the edges of these trees (which, obviously, are edges of the cube incident to white vertices) since the number of edges of a tree with v vertices is $v - 1$. Hence, except for the root, each vertex of a rooted tree is associated with the edge whose end-points are itself and its father. After degenerating all the edges of a tree, we degenerate the edge whose end-points are the root and a black vertex adjacent to it.

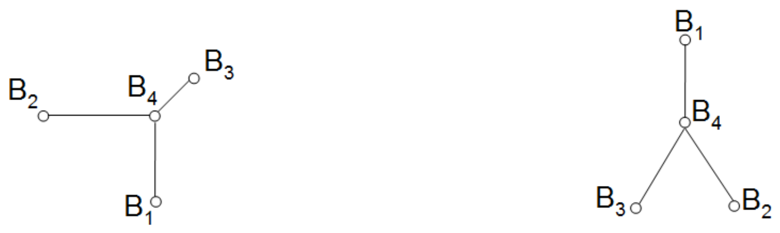


Fig. 3.24: On the left: the subgraph obtained by deleting the black vertices of the cube shown in Figure 3.23. On the right: its rooted spanning tree.

Remark 3.1.18. *In the procedure described previously, we have imposed that the root of each tree is adjacent to a black vertex of the cube. This condition is essential for obtaining the order relation since after degenerating the edges of the tree, it allows us that the root is still incident to an edge of the cube. This edge is the last one in degenerating. Let us observe that this condition is not satisfied in Figure 3.23, since the last edge in degenerating is incident to the white vertex B_4 which is not adjacent to any black vertex of the cube.*

On the other hand, the procedure assures that the degenerated edges are always edges of the cube. It is because they are edges of a spanning tree of a subgraph of the cube. In this way, we avoid situations as that shown in Figure 3.23.

In Figure 3.25, we show an order relation for degenerating the edges of the cubes which contain the pattern subsets $(V_3)_3$ and $(V_1)_1$, respectively.

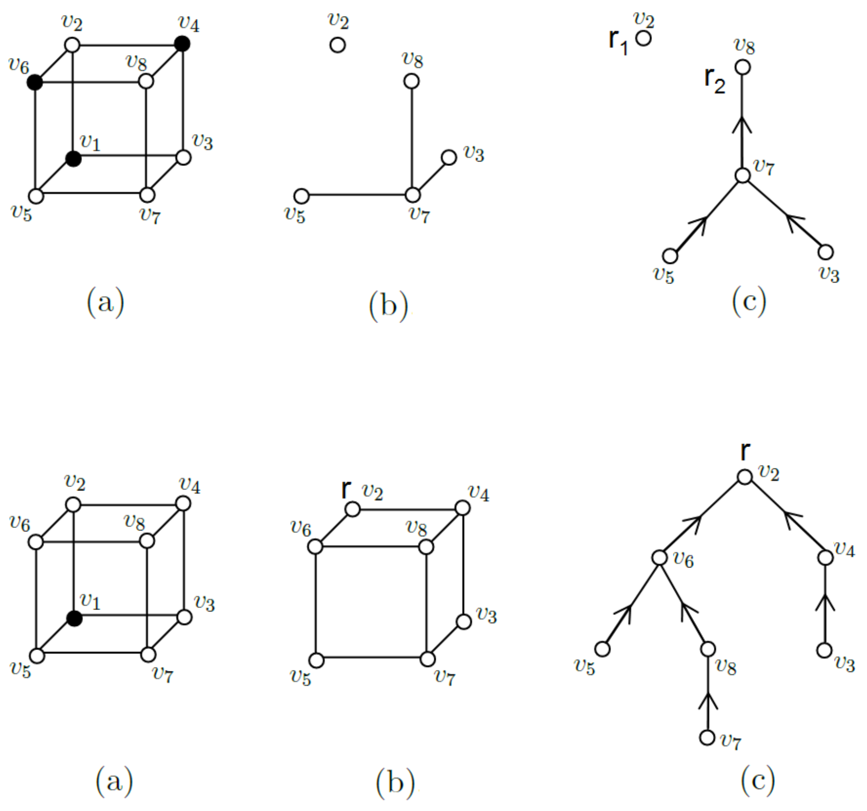


Fig. 3.25: On the top: (a) the pattern subset $(V_3)_3$; (b) the subgraph of the cube obtained by deleting the points of $(V_3)_3$ and the edges of the cube incident to these points; (c) the rooted spanning trees of the two connected components of the subgraph, respectively. On the bottom: (a) the pattern subset $(V_1)_1$; (b) the subgraph of the cube obtained by deleting the points of $(V_1)_1$ and the edges of the cube incident to these points; (c) the rooted spanning tree of the single connected component of the subgraph. The root of each tree is denoted by r and the arrows indicate how to degenerate the edges.

3.1.2.3 *Convex hull of a pattern subset*

We describe a procedure for constructing the convex hull of any pattern subset in $\mathbb{Z}^3$. These convex hulls allow us to determine (up to isometries) the cells (together with their boundary) which can be obtained by deforming a cube. The procedure is based on degenerating edges of the cube incident to white vertices.

As we commented at the beginning of Section 3.1.2, the degeneracy of an edge of the cube incident to a white vertex can lead to the degeneracy of faces and/or volumes contained in the cube. Below, we relate the existence of degenerated faces and/or volumes to the *star* of the degenerated edge and to that of the white vertex to treat. We recall that the *star of a cell c* is the set of cells whose boundary contains c.

Let $(V_c)_i$ be a pattern subset contained in the cube $C_{(V_c)_i}$, and let S be the subgraph of $C_{(V_c)_i}$ constructed by deleting the points of $(V_c)_i$. For each white vertex B of $C_{(V_c)_i}$, we consider the rooted spanning tree T of the connected component of S containing B, and the edge $e = AB$ of T whose end-points are B and its father. Let $S(B)$ be the star of the white vertex to treat, and let $S(e)$ be the star of the edge e. Next, we degenerate e.

1. Let f be a square face contained in $S(B)$.

   (a) If $f \notin S(e)$, then f is deformed into a face of four non-coplanar vertices. In this case, in order to avoid convexity problems related to this non-planar face, we attach the edge whose vertices are the other end-points of the edges of f incident to B. This edge allows us to cut up the non-planar face into two triangular faces sharing an edge. The shared edge is the attached edge. See Figure 3.26 for a pictorial description.

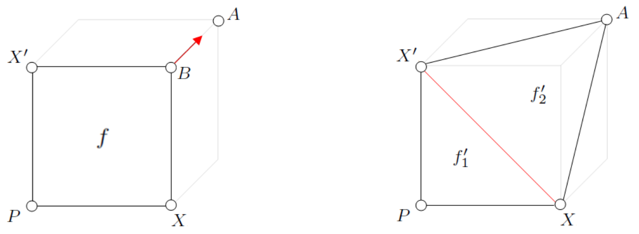


Fig. 3.26: The square face f is deformed into a non-planar face with four vertices, as a consequence of degenerating an edge which is not incident to f. To avoid convexity problems, the red edge is attached. This edge cuts up the non-planar face into two triangular faces, f_1' and f_2'.

   (b) If $f \in S(e)$, then f is deformed into a triangular face f'. The vertices of f' are those of f except B (see Figure 3.27).

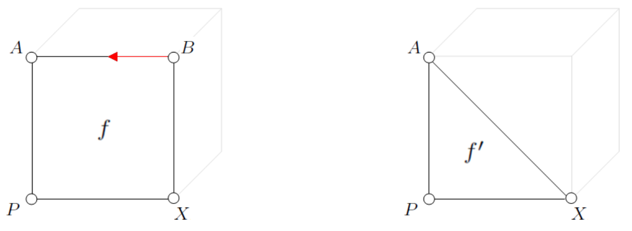


Fig. 3.27: The square face f is deformed into the triangular face f', as a consequence of degenerating an edge which is incident to f.

2. Let T be a triangular face contained in $S(B)$.

    (a) If $T \notin S(e)$, then T is deformed into another triangular face T'. The vertices of T' are those of T except B, which is replaced with the other end-point of e. An example of this case is shown in Figure 3.28.

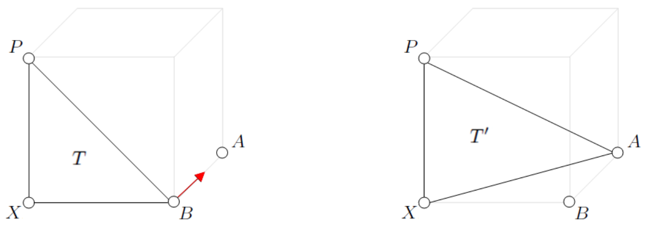


Fig. 3.28: The triangular face T is deformed into another triangular face T', as a consequence of degenerating an edge which is not incident to T.

    (b) If $T \in S(e)$, then T degenerates into an edge $e' \neq e$ incident to T. The end-points of e' are the vertices of T except B. The degenerated edge leads to a degenerated triangular face. This degenerated triangular face must be removed starting from the third edge $a \neq e, e'$ incident to T (see Figure 3.29).

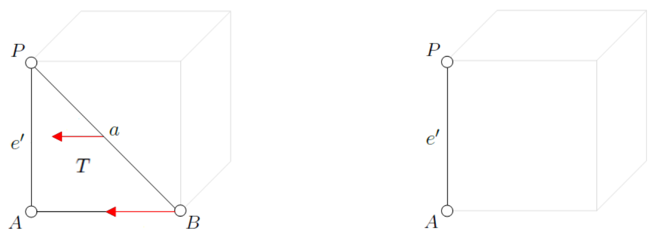


Fig. 3.29: The triangular face T degenerates into an edge e' which is incident to T. This degenerated face appears as a consequence of degenerating an edge of the cube which is incident to T. The degenerated face is removed starting from the edge a.

3. Let V be a volume contained in $C_{(V_c)_i}$, and let $\mathcal{F}$ be the set of faces of V.

   If there exists only one face $f \in \mathcal{F} - S(B)$, then V degenerates into f. The degenerated edge leads to a degenerated volume. This degenerated volume must be removed starting from a face incident to it (see Figures 3.30 and 3.31).

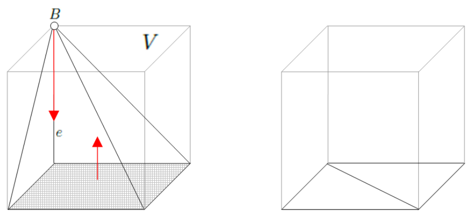


Fig. 3.30: The only face of the pyramid which does not belong to $S(B)$ is its square base, so the pyramid degenerates into it. This degenerated pyramid is removed starting from its square base.

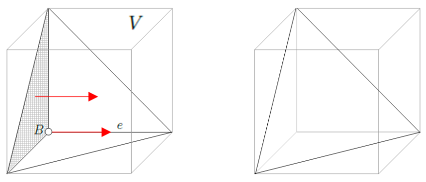


Fig. 3.31: There exists only one face of the tetrahedron which does not belong to $S(B)$, so the tetrahedron degenerates into it. This degenerated tetrahedron is removed starting from the only face contained in $S(B) - S(e)$.

As a consequence of these operations, two coplanar triangular faces sharing an edge may have appeared (see Figure 3.30). In this case, we must remove the shared edge. The two coplanar triangular faces become only one square face. Hence, the vertices of this square face are those of the two triangular faces.

Summarizing, the developed technique for constructing the convex hull of a pattern subset consists in: (1) attaching edges to convert non-planar faces with four vertices into two triangular faces sharing an edge; (2) studying the degenerated i–cells, for $i = 1, 2, 3$; and (3) removing edges for converting two coplanar triangular faces into only one square face.

In Algorithm 3.1.3, we define the procedure for: (1) deforming the unit cube into the convex hull of any pattern subset $(V_c)_i \subset \mathbb{Z}^3$; (2) constructing the cell defined by this pattern subset; and (3) computing its boundary.

Algorithm 3.1.3

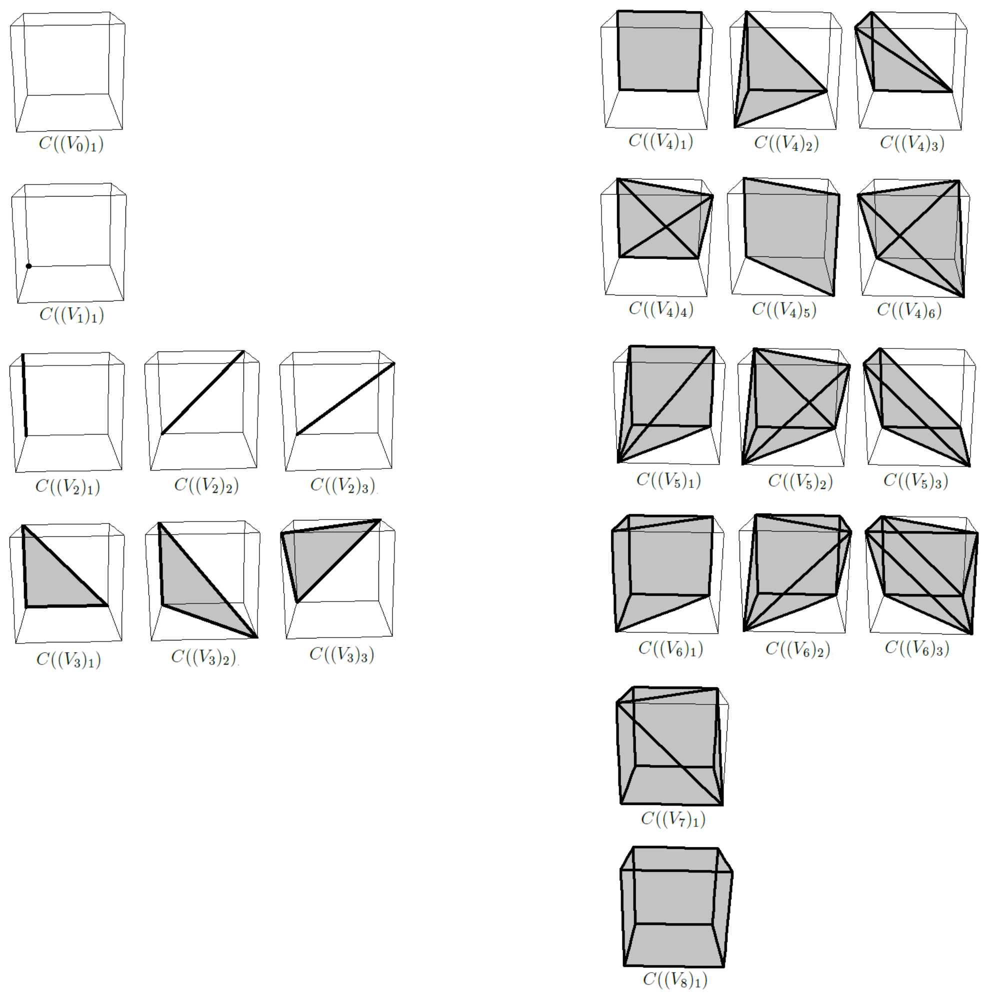
```
Input: the unit cube $C_V$.
        a pattern subset $(V_c)_i \subset C_V$ with $c$ points.
Output: convex hull of $(V_c)_i$.
begin
// $Ve$: vertices of $C_V$.
// $Ed$: edges of $C_V$.
// $Fa$: faces of $C_V$.
// $Vo$: volume of $C_V$.
// $S$: subgraph of $C_V$ constructed by deleting the points of $(V_c)_i$.
// $WV$: set of white vertices of $C_V$ ordered according to Section 3.1.2.2.
 1: $Ed' = Ed$
 2: for each $B \in WV$ ordered according to Section 3.1.2.2 do
 3:    • Construct the rooted spanning tree $T$ of the connected component of $S$ contain-
       ing $B$
 4:    • Degenerate the edge incident to $B$ and to its father in $T$
       {This edge degeneracy replaces $B$ with $A$, where $A$ is the father of $B$ in $T$.}
 5:    • Obtain the lists $Ve'$, $Ed'$, $Fa'$, $Vo'$ of vertices, edges, faces and volumes, respec-
       tively, by replacing $B$ with $A$
 6:    for each $f' \in Fa'$ do
 7:       if $f'$ is a face of 4 non-coplanar vertices then
 8:          • $Ed' = Ed' \bigcup \{XX'\}$, where $X, X' \in Ve'$ satisfy that $XA, X'A \in \partial(f')$
             {We attach an edge for cutting up the non-planar face into two triangular
             faces.}
 9:          • $Fa' = Fa' \bigcup \{XX'A\}$
10:          • $Fa' = Fa' \bigcup \{XX'P\}$, where $P \neq A$ is such that $P \in Ve'$ and it satisfies
             $XP, X'P \in \partial(f')$
             {We attach the two triangular faces.}
11:          • $Fa' = Fa' - \{f'\}$
             {We remove the non-planar face.}
12:       end if
13:       if $f'$ is a face degenerated into an edge $e' \in Ed'$ then
```

```
14:         $Fa' = Fa' - \{f'\}$
            {We remove $f$ starting from $a$, where $f$ is the triangular face which became
            $f'$ when $B$ was replaced with $A$, and $a \neq e, e'$ is the third edge of $f$.}
15:       **end if**
16:       **if** $v' \in Vo'$ is a volume degenerated into $f'$ **then**
17:         • $Vo' = Vo' - \{v'\}$
            {We remove $v$ starting from $g$, where $v$ is the volume which became $v'$ when
            $B$ was replaced with $A$, and $g$ is a face incident to $v$.}
18:       **end if**
19:       **if** $f' \neq f'' \in Fa'$ are two coplanar triangular faces sharing an edge $e'' \in Ed'$
          **then**
20:         **if** $\{f' + f''\} \notin Fa'$, where $f' + f''$ denotes the closure of the all the points
            inside of the convex hull of the vertices of $f'$ and $f''$ **then**
21:           • $Fa' = Fa' \bigcup \{f' + f''\}$
              {We attach the square face made up by the two coplanar triangular faces.}
22:         **end if**
23:         • $Fa' = Fa' - \{f'\}$
24:         • $Fa' = Fa' - \{f''\}$
25:         • $Ed' = Ed' - \{e''\}$
            {We remove the two coplanar triangular faces and the edge shared by both
            faces.}
26:       **end if**
27:     **end for**
28: **end for**
29: **return** $Ve', Ed', Fa'$
**end**
```

Remark 3.1.19. *Given a pattern subset $(V_c)_i$, Algorithm 3.1.3 deforms and degenerates the cells contained in the unit cube C_V for computing the convex hull of the points of $(V_c)_i$. In this way, the cell $C((V_c)_i)$ defined by $(V_c)_i$ is determined by the points inside of this convex hull. Moreover, the boundary of $C((V_c)_i)$ is computed in terms of the vertices, edges and faces of the convex hull. More concretely, $\partial(C((V_c)_i))$ is given by Ve', Ed', Fa'.*

By using as input the 22 pattern subsets shown in Table 3.5, Algorithm 3.1.3 returns the 22 pattern cells shown in Table 3.6. Let us note that 12 of them are 3–cells (see Figure 1 in [1]). More concretely, $C((V_4)_2), C((V_4)_3), C((V_4)_4), C((V_4)_6), C((V_5)_1), C((V_5)_2), C((V_5)_3), C((V_6)_1), C((V_6)_2), C((V_6)_3), C((V_7)_1), C((V_8)_1)$ are 3–cells.

Remark 3.1.20. *The pattern cells shown in Table 3.6 are stored in a look-up table. In the next section, this table is used to construct a cell complex from a 3–dimensional binary digital image.*


$C((V_0)_1)$
$C((V_4)_1)$ $C((V_4)_2)$ $C((V_4)_3)$
$C((V_1)_1)$
$C((V_4)_4)$ $C((V_4)_5)$ $C((V_4)_6)$
$C((V_2)_1)$ $C((V_2)_2)$ $C((V_2)_3)$
$C((V_5)_1)$ $C((V_5)_2)$ $C((V_5)_3)$
$C((V_3)_1)$ $C((V_3)_2)$ $C((V_3)_3)$
$C((V_6)_1)$ $C((V_6)_2)$ $C((V_6)_3)$
$C((V_7)_1)$
$C((V_8)_1)$

Tab. 3.6: Pattern cells defined by the pattern subsets shown in Table 3.5.

3.2 Construction of the cell complex

We present a procedure for constructing a cell complex from a given 3–dimensional binary digital image. The work is done in five steps (see Figure 3.32): (1) computing the subsets of points of the image; (2) associating each of these subsets with a pattern subset; (3) identifying the pattern cell defined by each of these pattern subsets; (4) inverting the isometry between each subset and its associated pattern subset. It allows us to obtain the cells of the cell complex. Let us note that a cell can appear several times, since it can be contained in more than one cube of the grid. In this sense, (5) the cell complex is constructed by attaching only one copy of each cell. Moreover, this cell complex can be simplified by removing the faces incident to two volumes, the edges incident to exactly two coplanar faces, and the vertices incident to exactly two collinear edges. Some of these simplifications are treated in Section 3.3.

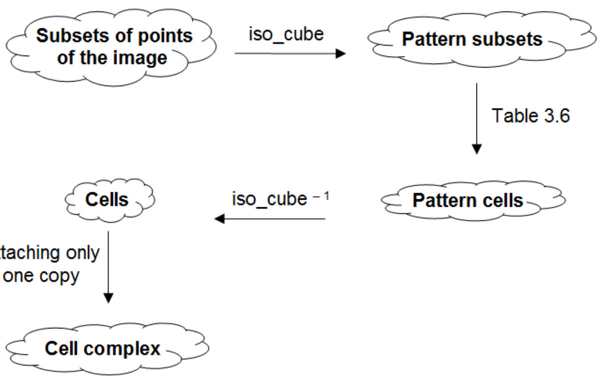


Fig. 3.32: The diagram shows the procedure which allows us to construct the cell complex from the image.

3.2.1 *Subsets of points of an image*

This subsection is devoted to Stages (1) and (2) working out only the vertex level. Given a binary digital image on a dual grid, we want to associate each subset of points of the image with a pattern subset. The association is done by *localizing* and *classifying* the subsets of black points of the image in each cube of the grid. The step of *localization* consists in scanning each cube of the grid and checking which is the subset of points corresponding to its vertex set. The step of *classification* consists in finding the pattern subsets of the image.

3.2.1.1 *Localizing subsets of points of an image*

Let I be a binary digital image which consists of r black points $v_1, ..., v_r$ on a dual grid G made up by $m_1 \times m_2 \times m_3$ cubes, with $0 \leq r \leq (m_1 + 1)(m_2 + 1)(m_3 + 1)$. A naive algorithm of localization (as Algorithm 3.2.4) allows us to scan each cube of the grid and check which is the subset of points corresponding to its vertex set.

Algorithm 3.2.4

Input: black points $\{v_1, ..., v_r\}$ of a dual image I.
dual grid $G = \{C_1, ..., C_{m_1 m_2 m_3}\}$.
Output: subsets of points of the image.

```
begin
// $V(I)$: empty set to store the lists of points of the image.
1: for each cube $C_i \subset G$ do
2:     Let $L$ be the list of black vertices of $C_i$
3:     $V(I) = V(I) \bigcup \{L\}$
4: end for
5: return $V(I)$
end
```

In Example 3.2.1, Algorithm 3.2.4 is used to localize the subsets of points of a binary digital image.

Example 3.2.1. *Let* $\{\{0,0,0\},\{0,0,1\},\{0,0,2\},\{0,0,4\},\{0,1,0\},\{0,1,1\},\{0,1,3\},\{0,1,4\},\{0,2,1\},\{0,2,2\},\{0,2,4\},\{0,3,1\},\{0,3,2\},\{0,3,3\},\{0,3,4\},\{0,4,0\},\{0,4,1\},\{0,4,2\},\{0,4,3\},\{1,0,0\},\{1,0,1\},\{1,0,2\},\{1,0,3\},\{1,0,4\},\{1,1,1\},\{1,1,2\},\{1,1,3\},\{1,2,0\},\{1,2,1\},\{1,2,3\},\{1,2,4\},\{1,3,0\},\{1,3,1\},\{1,3,2\},\{1,3,3\},\{1,3,4\},\{1,4,0\},\{1,4,2\},\{2,0,1\},\{2,0,3\},\{2,0,4\},\{2,1,0\},\{2,1,1\},\{2,1,2\},\{2,1,3\},\{2,2,1\},\{2,2,2\},\{2,2,4\},\{2,3,0\},\{2,3,1\},\{2,3,3\},\{2,3,4\},\{2,4,1\},\{2,4,2\},\{2,4,3\},\{3,0,0\},\{3,0,1\},\{3,0,3\},\{3,0,4\},\{3,1,0\},\{3,1,2\},\{3,1,3\},\{3,1,4\},\{3,2,0\},\{3,2,1\},\{3,2,2\},\{3,2,3\},\{3,2,4\},\{3,3,0\},\{3,3,2\},\{3,3,3\},\{3,3,4\},\{3,4,0\},\{3,4,1\},\{3,4,2\},\{3,4,3\},\{3,4,4\},\{4,0,1\},\{4,0,2\},\{4,0,3\},\{4,0,4\},\{4,1,0\},\{4,1,1\},\{4,1,2\},\{4,1,3\},\{4,1,4\},\{4,2,0\},\{4,2,2\},\{4,2,4\},\{4,3,0\},\{4,3,2\},\{4,4,0\},\{4,4,1\},\{4,4,3\},\{4,4,4\}\}$ *be the set of the 95 black points of a dual grid G made up by* $4 \times 4 \times 4$ *cubes.*

Algorithm 3.2.4 scans each of the 64 cubes of the grid and checks which is the subset of points corresponding to its vertex set. In Figure 3.33, we show several stages of the cube scanning.

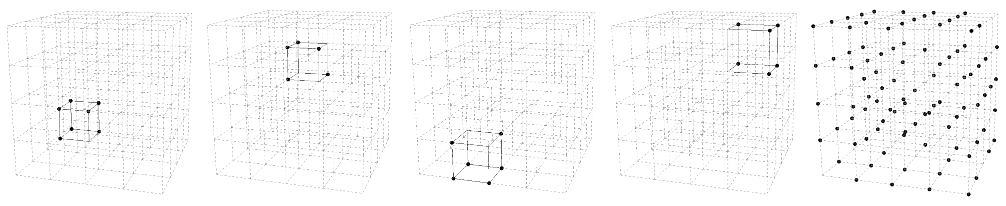


Fig. 3.33: The first four pictures correspond to the scanning of the cubes 1, 7, 53 and 64 of the grid, respectively. The last one represents the binary digital image defined by the 95 black points in Example 3.2.1.

Remark 3.2.2. *Algorithm 3.2.4 splits the set of black points of a binary digital image. The method for associating each of these subsets with one of the pattern subset shown in Table 3.5 is explained next.*

3.2.1.2 Classifying subsets of points of an image

Algorithm 3.1.2 at pages 47–48 computes the action of the group of isometries of a cube on the subsets of points. This allows us to characterize the subsets of points in two disjoint families: (a) one consisting of the pattern subsets; and (b) other one with the rest of subsets. Note that each subset of type (b) is isometric to a pattern subset, i.e. to a subset of type (a). In this way, given $V(I)$ as the set containing the subsets of points of an image I, a naive algorithm (as Algorithm 3.2.5) allows us to associate each subset contained in $V(I)$ with a pattern subset, giving the set of the pattern subsets of I.

Algorithm 3.2.5

```
Input: $V(I)$: subsets of points of a binary digital image.
       $PS$: pattern subsets.
Output: pattern subsets of the image.
begin
 1: for $X \in V(I)$ do
 2:    if $Y \in PS \bigcap iso\_cube(X)$ then
 3:       Replace $X$ with $Y$ in $V(I)$
 4:       Save $\sigma \in iso\_cube$ such that $\sigma(X) = Y$
 5:    end if
 6: end for
 7: return $V(I)$
end
```

Example 3.2.3. *Algorithm 3.2.5 replaces each subset of points of the image shown in Example 3.2.1 with a pattern subset, giving the set of the pattern subsets of the image. In Figure 3.34, the pattern subsets of this image are shown for the cubes 1, 7, 53 and 64 of the grid.*

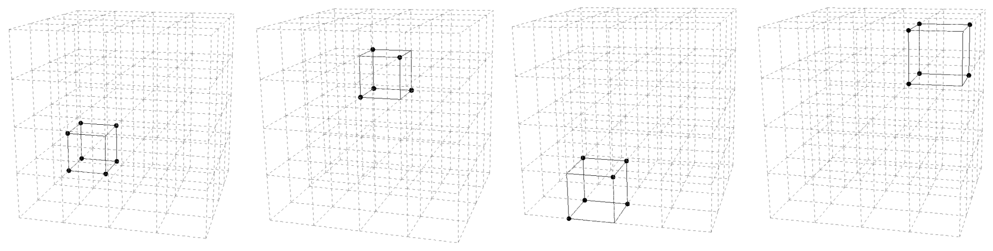


Fig. 3.34: Pattern subsets of the image shown in Example 3.2.1 for the cubes 1, 7, 53 and 64 of the grid, respectively.

Theorem 3.2.4. *Any 3–dimensional binary digital image on a dual grid is made up by combining (up to isometries) the 22 pattern subsets shown in Table 3.5, taking into account that it is not necessary to use all them and any of them can be used more than once.*

The results shown in Theorem 3.2.4 are determined by Algorithms 3.2.4 and 3.2.5, since they allow us to find the pattern subsets of any given image.

3.2.2 Cells of the cell complex

This subsection is devoted to Stages (3), (4) and (5) working out only the cell level. The cells (together with their boundary) of the cell complex can be obtained following these two steps: (a) *identifying* the pattern cells defined by the pattern subsets of the image; and (b) *inverting the isometry* from each subset of points of the image to its associated pattern subset. Finally, the cell complex is constructed by attaching only one copy of each of these cells along their boundaries.

3.2.2.1 Pattern cells of the cell complex

We identify the cells defined by each of the pattern subsets of the image according to Table 3.6.

Remark 3.2.5. *By identifying each subset with a cell, we obtain the pattern cells of the cell complex.*

In Figure 3.35, we present the cells defined by some of the pattern subsets of the image shown in Example 3.2.1.

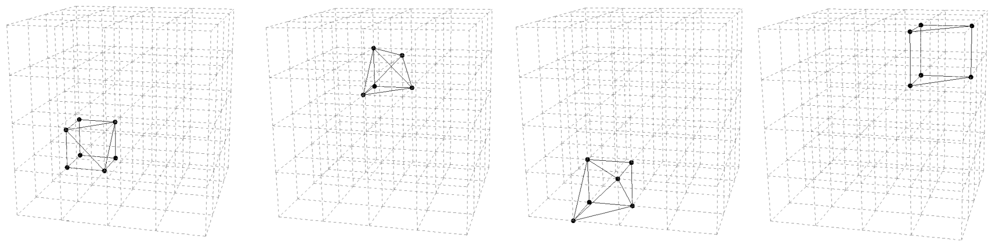


Fig. 3.35: Pattern cells (together with their boundary) defined by the pattern subsets of the image shown in Example 3.2.1 for the cubes 1, 7, 53 and 64 of the grid, respectively.

3.2.2.2 Inverting the isometries

Algorithm 3.2.5 replaces each subset X of points of the image with a pattern subset Y. This subset Y is the image of X under the action of an element σ of the group of isometries of a cube; i.e. each vertex $p \in X$ is replaced with a vertex $p' = \sigma(p) \in Y$.

Let $C(Y)$ be the pattern cell corresponding to the pattern subset Y. This cell consists of (1) the inside of the convex hull of the points of Y, and (2) the boundary of this convex hull. In this way, if we replace every point with its pre-image under σ, then we obtain the inside of the convex hull of the points of X, and the boundary of this convex hull; i.e. the cell $C(X)$ corresponding to the subset X.

Remark 3.2.6. *By applying this procedure to each subset of points of the image, we obtain the cells (together with their boundary) defined by these subsets. The cell complex is constructed by attaching only a copy of each of these cells along their boundaries (see Figure 3.36 for a simple example). The results shown in Theorem 3.2.7 are determined by this procedure.*

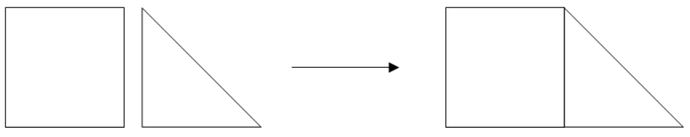


Fig. 3.36: By attaching a square face to a triangular face along their common edge, we construct a cell complex.

Theorem 3.2.7. *The cell complex constructed from a given 3–dimensional binary digital image on a dual grid is made up by combining (up to isometries) the 22 pattern cells shown in Table 3.6, taking into account that it is not necessary to use all them and any of them can be used more than once.*

In Figure 3.37, we show the cells (together with their boundary) resulting of inverting the isometries which associate the subsets of points of the image shown in Example 3.2.1 with the pattern subsets contained in the cubes 1, 7, 53 and 64 of the grid, respectively. Additionally, we show the cell complex constructed from this image.

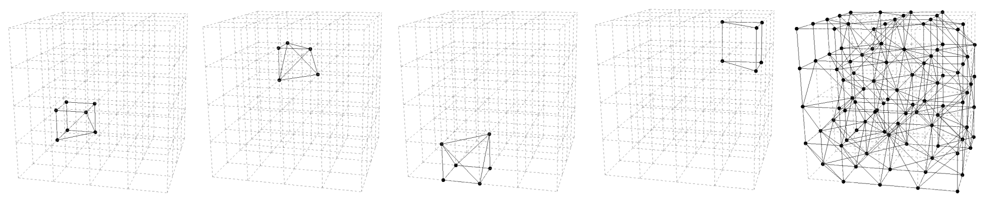


Fig. 3.37: The first four pictures correspond to the cells (together with their boundary) defined by the subsets of points of the image shown in Example 3.2.1 for the cubes 1, 7, 53 and 64 of the grid, respectively. The last one represents the cell complex.

3.3 Simplification of the cell complex

We define an algorithm (Algorithm 3.3.6) for simplifying the cell complex constructed from a given 3–dimensional binary digital image. The simplification consists in removing the 2–cells of the complex incident to two 3–cells. More concretely, the *simplified complex* is given by: (a) the 0,1,2–cells of the initial complex which are not incident to a 3–cell; and (b) the set of 2–cells (together with their boundary) incident to exactly one of the 3–cells of the initial complex.

Algorithm 3.3.6

```
Input: $C_i(V(I))$: $i$–cells of the cell complex constructed from the image $I$, for $i = 0,1,2,3$.
       $\partial(C_3(V(I)))$: boundary of the cells contained in $C_3(V(I))$.
Output: simplified cell complex.
begin
// Bord: empty list to store the cells of the simplified complex.
1: for $i \in \{0,1,2\}$ do
2:    for each $i$–cell $c \in C_i(V(I))$ do
3:       if $c$ is not incident to a 3–cell then
4:          $Bord = Bord \bigcup \{c\}$
5:       else
6:          if $i \neq 2$ or $c$ is incident to exactly one 3–cell then
7:             $Bord = Bord \bigcup \{c\} \bigcup \{\partial c\}$
8:          end if
9:       end if
```

```
10:     end for
11: end for
12: return Bord
end
```

Example 3.3.1. *Algorithm 3.3.6 is applied to the cell complex shown in Figure 3.37. This algorithm determines that all the 0,1,2–cells are incident to a 3–cell, and it computes the 174 2–cells incident to exactly one of the 3–cells of the complex (see Figure 3.38).*

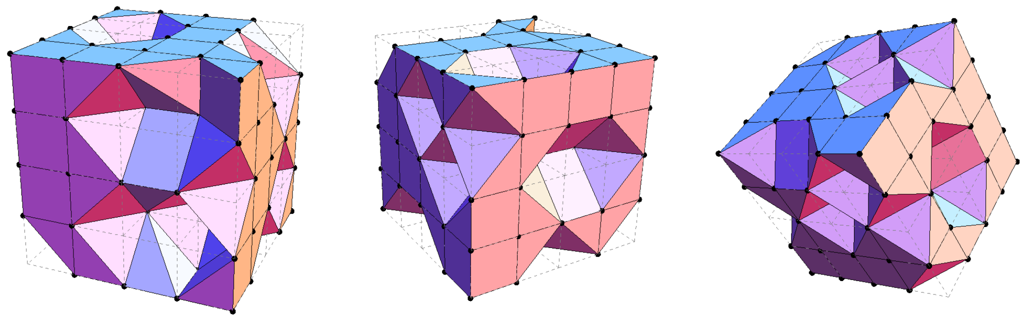
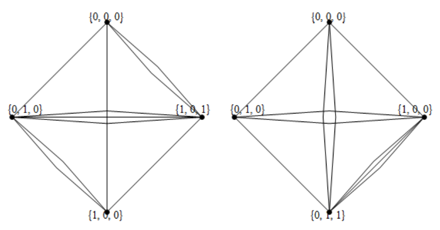


Fig. 3.38: Three viewpoints of the simplified cell complex obtained from the cell complex shown in Figure 3.37.

3.4 Comparison with Kenmochi et al. method

The main difference between the results obtained by Kenmochi et al. and our results is the number of configurations of white and black voxels on a cube. They consider that two configurations are the same if one is a rotation of the other one; whereas we consider that two configurations are the same if one is an isometry of the other one. In this way, Kenmochi et al. obtain 23 different configurations (see Table 2.8 at page 24); whereas we obtain 22 pattern subsets (see Table 3.5 at page 49). Moreover, each configuration in Table 2.8 corresponds (up to rotations) to a pattern subset in Table 3.5, except those with four black voxels. In this case, Kenmochi et al. obtain 7 configurations; whereas we obtain 6 pattern subsets. More concretely, the configurations P4c and P4d in Table 2.8 coincide (up to isometries) with the pattern subset $C((V_4)_3)$ in Table 3.5. Kenmochi et al. consider that the configurations P4c and P4d are different because there does not exist a rotation which sends P4c into P4d; whereas we consider that both configurations are identical because there exists a reflection from P4c into P4d. In Figure 3.39, we can see that the multi-graphs associated with P4c and P4d, respectively, are isomorphic.



{1, 4, 2, 3}

Fig. 3.39: On the left: multi-graphs associated with P4c and P4d, respectively. On the right: the isomorphism existing between both multi-graphs.

Consequently, Kenmochi et al. set down that a 3–dimensional binary digital image I can be made up (up to rotations) by both configurations P4c and P4d; whereas we asserts that I can only be made up (up to isometries) by one of the two configurations. In Example 3.4.1, we show this fact for a simple image.

Example 3.4.1. *Let* $\{\{0,0,0\},\{0,0,1\},\{0,2,1\},\{1,1,0\},\{1,1,1\},\{1,2,1\},\{2,0,1\},\{2,1,1\},\{2,2,0\},\{3,3,4\},\{3,3,5\},\{3,4,3\},\{3,4,4\},\{3,5,4\},\{4,4,3\},\{4,4,5\},\{4,5,3\},\{4,5,4\},\{5,0,0\},\{5,0,4\},\{5,0,5\},\{5,1,0\},\{5,1,5\},\{5,3,3\},\{5,3,4\},\{5,3,5\},\{5,4,3\},\{5,4,4\},\{5,5,0\},\{5,5,3\},\{5,5,4\}\}$ *be the set of the 31 black points of a dual grid of size* $6 \times 6 \times 6$.

On the first row in Figure 3.40, we show the scanning of the cubes 6, 26 and 93, and the representation of the image. On the second row in Figure 3.40, we show the configurations of Table 2.8 associated with the cubes 6, 26 and 93, respectively. Finally, on the third row in Figure 3.40, we show the pattern subsets of Table 3.5 associated with the cubes 6, 26 and 93, respectively.

Let us observe that Kenmochi et al. associate different configurations of Table 2.8 with the cubes 6 and 26 of the grid; whereas our method associates the same pattern subset of Table 3.5 with these cubes.

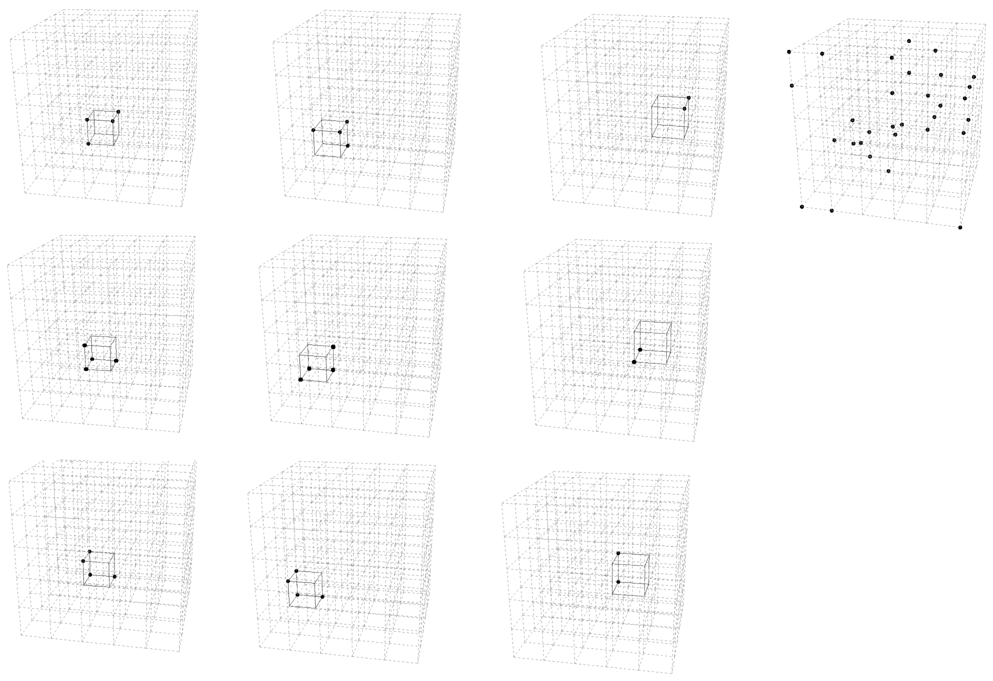


Fig. 3.40: On the first row: the first three pictures show the scanning of the cubes 6, 26 and 93, respectively; the fourth one shows the image. On the second row: configurations of Table 2.8 associated with the cubes 6, 26 and 93, respectively. On the third row: pattern subsets of Table 3.5 associated with the cubes 6, 26 and 93, respectively.

Another difference between Kenmochi et al. method and our method lies in the simplification of the cell complex obtained from the image. Kenmochi et al. construct the simplified complex simply from the 2–cells (together with their boundary) incident to exactly one of the 3–cells of the initial complex; whereas we also attach the 0,1,2–cells of the initial complex which are not incident to a 3–cell. See Figure 3.41 for an example.

Let us note that this last difference is question of choosing one or another way of simplifying the complex obtained from the image.

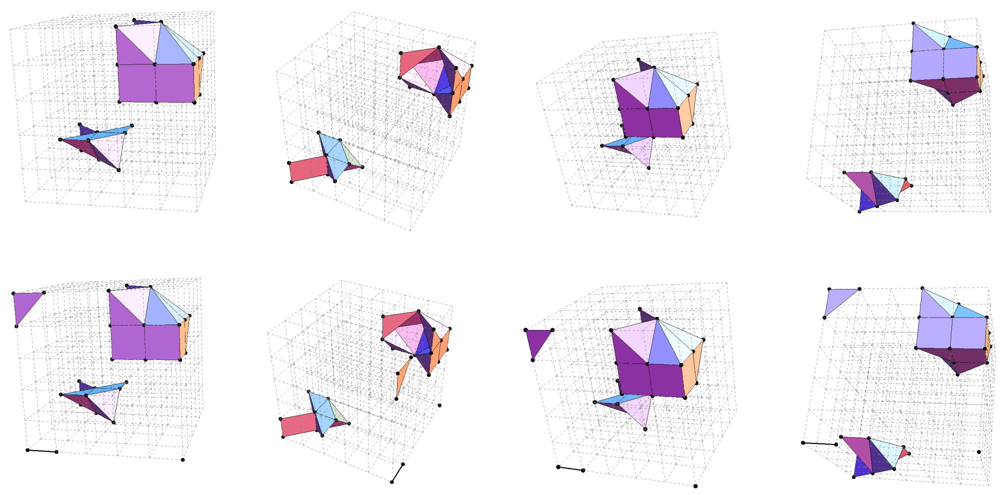


Fig. 3.41: On the top: different viewpoints of the simplified complex obtained by Kenmochi et al. from the image shown in Example 3.4.1. On the bottom: different viewpoints of the simplified complex obtained by Algorithm 3.3.6 from the image shown in Example 3.4.1.

4. EXTRACTING 4–DIMENSIONAL CELL COMPLEXES FROM BINARY DIGITAL IMAGES

In this chapter, we generalize to dimension four the work developed in Chapter 3. In this way, we use topological tools for constructing cell complexes from 4–dimensional binary digital images. Analogously, a *4–dimensional binary digital image* can be defined as a discrete subset of white and black points on a dual grid made up by 4–cubes. The black points are the 0–cells of a cell complex. By generalizing the algorithmic procedure shown in Chapter 3, every cell of the complex is constructed by deforming a 4–cube of the dual grid, in such a way that the cell is defined by the black points of the 4–cube. Finally, we keep the 0,1,2,3–cells of the complex which are not incident to a 4–cell, together with the 3–cells (and their boundary) which are incident to exactly one 4–cell of the complex.

This chapter has a structure similar to that of Chapter 3. In Section 4.1, we construct a look-up table containing (up to isometries) all the cells which can be obtained by deforming a 4–cube; in Section 4.2, we present a procedure for constructing the cell complex; and finally, in Section 4.3, we conceive an algorithm for simplifying the complex.

4.1 Look-up table construction

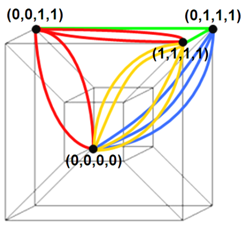
We construct all the cells which can be obtained (up to isometries) by deforming a 4–cube: (a) first, we determine the vertices which define these cells, they are the *pattern subsets*, and we store them in a look-up table (Appendix A); and (b) then, we compute the convex hull of each pattern subset. It allows us to determine the *pattern cells*, which are stored in another look-up table (Appendix B).

4.1.1 Pattern subsets

As we have commented previously, the first step of the procedure consists in computing (up to isometries) the different subsets of vertices of a 4–cube. There exist $2^{16} = 65536$ subsets of points which can be constructed from the vertices of a 4–cube. We extend to dimension four the two techniques described in Chapter 3 for computing the isometric subsets.

4.1.1.1 A first solution

Analogously to Chapter 3, a first idea consists in associating each subset with a multi-graph, in such a way that two subsets identified with a same pattern subset are associated with isomorphic multi-graphs. Let us note that the multi-graph associated with a subset of vertices of a 4–cube can have up to four edges between each pair of vertices. In Figure 4.1, we show the multi-graph associated with the subset $S = \{(0, 0, 0, 0), (0, 0, 1, 1), (0, 1, 1, 1), (1, 1, 1, 1)\}$ of vertices of the unit 4–cube, latter is represented by a Schlegel diagram.



$$\begin{array}{c|cccc} & (0,0,0,0) & (0,0,1,1) & (0,1,1,1) & (1,1,1,1) \\ (0,0,0,0) & 0 & 2 & 3 & 4 \\ (0,0,1,1) & 2 & 0 & 1 & 2 \\ (0,1,1,1) & 3 & 1 & 0 & 1 \\ (1,1,1,1) & 4 & 2 & 1 & 0 \end{array}$$

Fig. 4.1: The vertices of the multi-graph associated with $S = \{(0, 0, 0, 0), (0, 0, 1, 1), (0, 1, 1, 1), (1, 1, 1, 1)\}$ are the points of S and the element m_{ij} of its adjacency matrix coincides with the number of different coordinates between the vertices p_i and p_j.

By generalizing the notion of *isomorphic subsets* to subsets of points constructed starting from the vertices of a 4–cube, we can associate isomorphic subsets with isomorphic multi-graphs, and prove an analogous result to Theorem 3.1.2 established in Chapter 3.

Theorem 4.1.1. *Let S and S' be two subsets with $8 \leq c \leq 16$ points. S and S' are isomorphic subsets if and only if there exists a linear isometry $f : \mathbb{R}^4 \to \mathbb{R}^4$ which sends S into S' and the 4–cube HC_S containing S into the 4–cube $HC_{S'}$ containing S'.*

Proof. $\Rightarrow$ Starting from $S = \{p_1, ..., p_c\}$ (resp. $S' = \{q_1, ..., q_c\}$) we construct the vector space generated by $\{p_i - p_1\}_{2 \leq i \leq c}$ (resp. $\{q_i - q_1\}_{2 \leq i \leq c}$).

– If this vector space has four linearly independent vectors, then it will be a basis of $\mathbb{R}^4$. Therefore we can extend the isomorphism f between the multi-graphs G_S and $G_{S'}$ to an isometry from $\mathbb{R}^4$ to itself which sends S into S'. Moreover, f also sends HC_S into $HC_{S'}$. It suffices (1) to choose four vertices a_2, a_3, a_4, a_5 of HC_S such that $d(a_i, p_1) = 1$ for

$i = 2, 3, 4, 5$, (2) to write $p - p_1$ as a finite linear combination of the basis $\{a_2 - p_1, a_3 - p_1, a_4 - p_1, a_5 - p_1\}$, with $p \in HC_S$, and (3) to apply f for proving that $f(p)$ is a vertex of $HC_{S'}$.

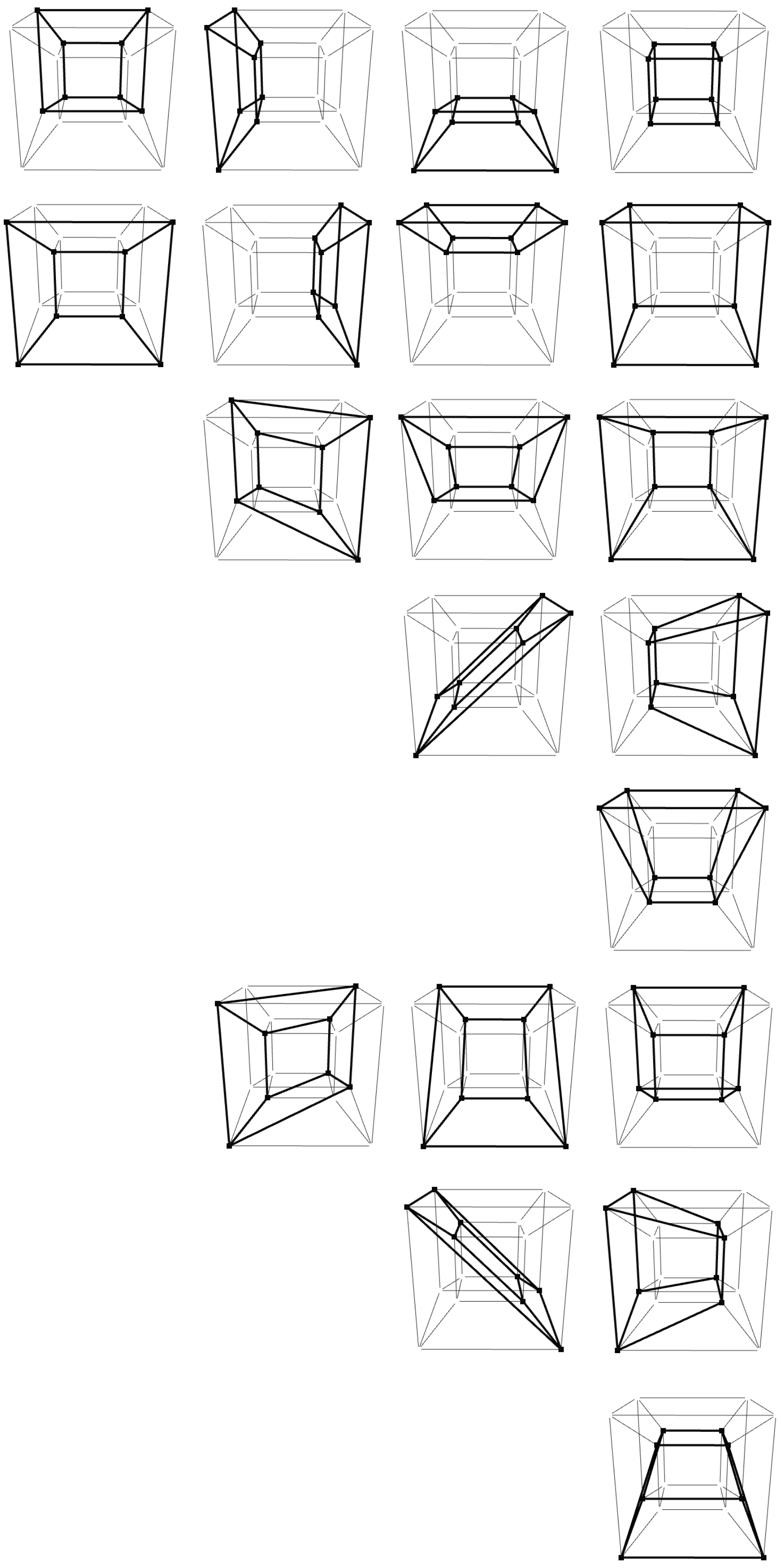
– If only three vectors of the vector space generated by $\{p_i - p_1\}_{2 \leq i \leq 8}$ (resp. $\{q_i - q_1\}_{2 \leq i \leq 8}$) are linearly independent, then S (resp. S') is made up by eight cospatial points. In a 4–cube, there exist two types of 8–tuple of cospatial vertices, those corresponding to vertices of one of the eight cubes of the 4–cube, and those not corresponding to vertices of one of the eight cubes of the 4–cube. The elements in the first group are called *outer cubic volumes*, whereas those in the second group are called *inner cubic volumes* (see Table 4.1). Taking into account that the points of S and S' have the same pairwise distances, we can suppose that the points of S and S' are the vertices of an outer (resp. inner) cubic volume contained in a 4–cube (see Remark 4.1.2 for more details). In this case, we must choose a ninth vertex $p \in HC_S - \{S\}$ (resp. $q \in HC_{S'} - \{S'\}$) which allows us to construct a basis of $\mathbb{R}^4$. Therefore we can extend the isomorphism f between the multi-graphs G_S and $G_{S'}$ to an isometry from $\mathbb{R}^4$ to itself which sends S into S'. Moreover, f also sends HC_S into $HC_{S'}$ since f sends nine vertices of HC_S into nine vertices of $HC_{S'}$.

$\Leftarrow$ If there exists an isometry σ which sends S into S' and HC_S into $HC_{S'}$, then σ preserves the distance between each pair of points of S and S'. Consequently, σ preserves the number of edges between each pair of vertices of the multi-graphs G_S and $G_{S'}$. Therefore these two multi-graphs are isomorphic to each other. Hence, S and S' are isomorphic.

□

Remark 4.1.2. *The unit 4–cube has twenty 8–tuple of cospatial vertices, which correspond to vertices of cubic volumes. More concretely, the eight outer cubic volumes are given by the intersection of the unit 4–cube with the hyperplanes $X = 0$, $Y = 0$, $Z = 0$, $T = 0$, $X = 1$, $Y = 1$, $Z = 1$, $T = 1$, respectively; and the twelve inner cubic volumes are given by the intersection of the unit 4–cube with the hyperplanes $X = Y$, $X = Z$, $X = T$, $Y = Z$, $Y = T$, $Z = T$, $X + Y = 1$, $X + Z = 1$, $X + T = 1$, $Y + Z = 1$, $Y + T = 1$, $Z + T = 1$, respectively. In Table 4.1*[1] *the twenty cubic volumes contained in a 4–cube are shown.*

1. screenshots of "**Hypercube**" software developed by Régis Meyssonnier in his Final Master Project supervised by Prof. Jean-Luc Mari from University of Marseille (see [29])



Tab. 4.1: The first two rows show the eight outer cubic volumes contained in a 4–cube. The last six ones show the inner ones.

Remark 4.1.3. *Let us note that Theorem 4.1.1 is not true if the subsets have 7 or less points. It suffices to choose* $S = \{\{0,0,1,1\},\{0,1,0,1\},\{1,0,0,1\},\{1,1,1,1\}\}$ *and* $S' = \{\{0,1,1,1\},\{1,0,1,1\},\{1,1,0,1\},\{1,1,1,0\}\}$. *The multi-graphs* G_S *and* $G_{S'}$ *associated with* S *and* S'*, respectively, are isomorphic, but there does not exist a linear isometry which sends* S *and* S' *and* HC_S *into* $HC_{S'}$ *(see Figure 4.2).*

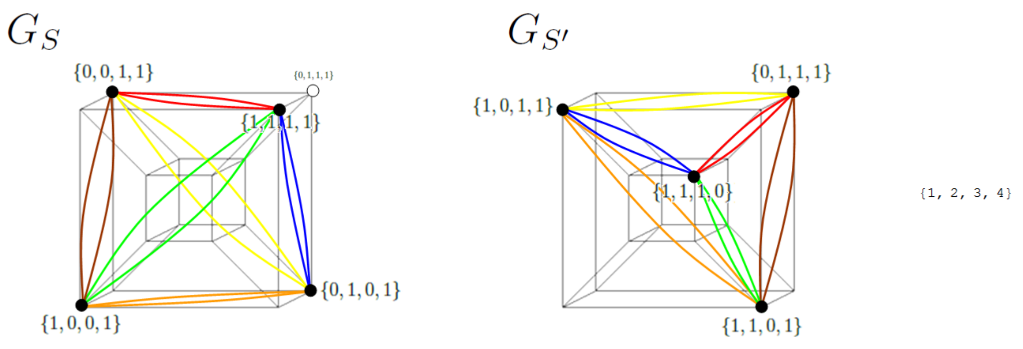


Fig. 4.2: The multi-graph G_S is isomorphic to the multi-graph $G_{S'}$, but the isometry which sends S into S' does not send HC_S into $HC_{S'}$. It suffices to check that the white vertex $(0,1,1,1) \in HC_S$ is not sent to any white vertex in $HC_{S'}$ by the isometry.

Taking into account the previous results, a generalization of Algorithm 3.1.1 in Chapter 3 is conceived. By using as input the set $(V, \prec)$ of vertices of the unit 4–cube arranged on an order relation $\prec$, for $8 \leq c \leq 16$, this algorithm: (a) constructs an ordered set $(V_c, \prec)$ containing the $C(16, c)$ subsets with c points; (b) associates each subset $(V_c)_i \subset V_c$ with a multi-graph $(G_{V_c})_i$; and (c) checks if there exists an isomorphism between each pair $(G_{V_c})_{i_1}, (G_{V_c})_{i_2}$ of multi-graphs associated with two subsets $(V_c)_{i_1}, (V_c)_{i_2} \subset V_c$ which satisfy $(V_c)_{i_1} \prec (V_c)_{i_2}$. In Example 4.1.4, we describe with detail the procedure for $c = 8$.

Example 4.1.4. *Let* $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}, v_{12}, v_{13}, v_{14}, v_{15}, v_{16}\} = \{\{0,0,0,0\},\{0,0,0,1\},\{0,0,1,0\},\{0,0,1,1\},\{0,1,0,0\},\{0,1,0,1\},\{0,1,1,0\},\{0,1,1,1\},\{1,0,0,0\},\{1,0,0,1\},\{1,0,1,0\},\{1,0,1,1\},\{1,1,0,0\},\{1,1,0,1\},\{1,1,1,0\},\{1,1,1,1\}\}$ *be the set of vertices of the unit 4–cube arranged on lexicographic order* $(\prec_{lex})$.

First, we consider the ordered set $(V_8, \prec_{lex})$ *containing the* $C(16, 8)$ *subsets with eight points of* V.

Next, for $i = 1, \ldots, C(16, 8)$ *each subset* $(V_8)_i \subset V_8$ *is associated with a multi-graph* $(G_{V_8})_i$. *The vertices of* $(G_{V_8})_i$ *are the points of* $(V_8)_i$ *and the number of edges between two vertices* $v_k, v_l \in (G_{V_8})_i$ *is determined by the* *number of different coordinates between v_k and v_l. Some of these multi-graphs are shown in Figure 4.3.*

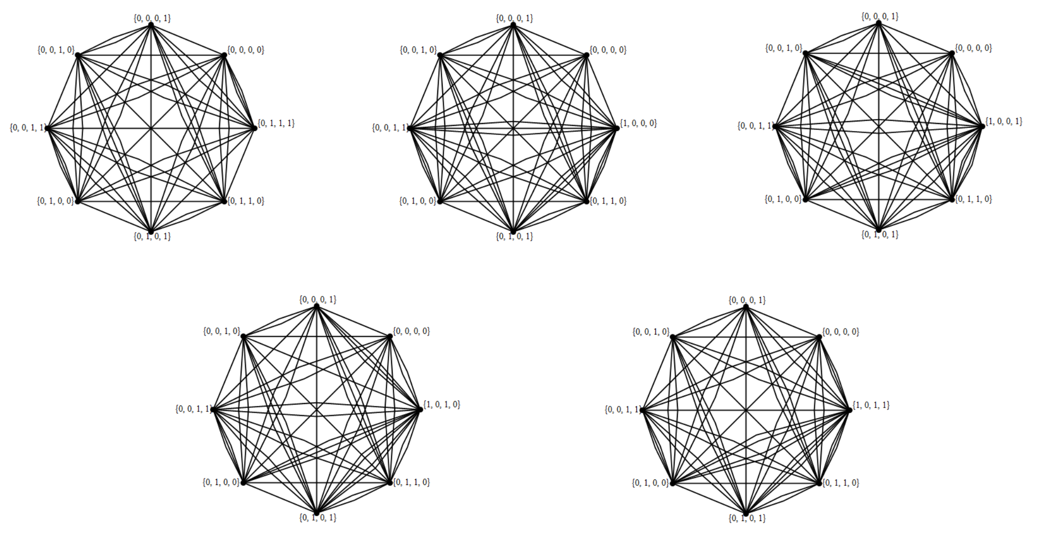
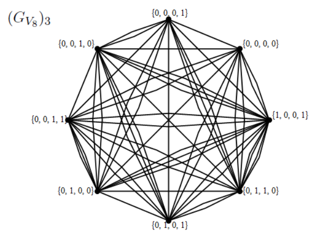
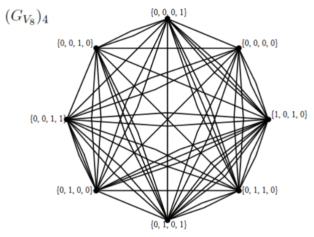
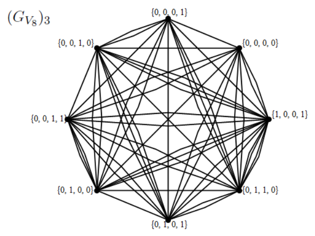
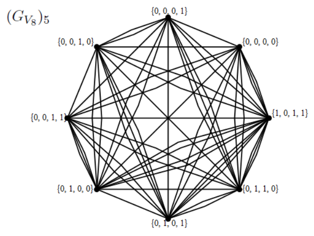


Fig. 4.3: Multi-graphs associated with some of the subsets of V_8.

Finally, it is checked if there exists an isomorphism between each pair $(G_{V_8})_{i_1}, (G_{V_8})_{i_2}$ of multi-graphs associated with two subsets $(V_8)_{i_1}, (V_8)_{i_2} \subset V_8$ which satisfy $(V_8)_{i_1} \prec_{lex} (V_8)_{i_2}$. In Table 4.2, we show the isomorphisms between some pairs of these multi-graphs.

 $(G_{V_8})_3$
{0, 0, 0, 1}
{0, 0, 1, 0}
{0, 0, 0, 0}
{0, 0, 1, 1}
{1, 0, 0, 1}
{0, 1, 0, 0}
{0, 1, 1, 0}
{0, 1, 0, 1}
	 $(G_{V_8})_4$
	{0, 0, 0, 1}
	{0, 0, 1, 0}
	{0, 0, 0, 0}
	{0, 0, 1, 1}
	{1, 0, 1, 0}
	{0, 1, 0, 0}
	{0, 1, 1, 0}
	{0, 1, 0, 1}
		{1, 3, 2, 4, 5, 7, 6, 8}

 $(G_{V_8})_3$
{0, 0, 0, 1}
{0, 0, 1, 0}
{0, 0, 0, 0}
{0, 0, 1, 1}
{1, 0, 0, 1}
{0, 1, 0, 0}
{0, 1, 1, 0}
{0, 1, 0, 1}
	 $(G_{V_8})_5$
	{0, 0, 0, 1}
	{0, 0, 1, 0}
	{0, 0, 0, 0}
	{0, 0, 1, 1}
	{1, 0, 1, 1}
	{0, 1, 0, 0}
	{0, 1, 1, 0}
	{0, 1, 0, 1}
		{}

Tab. 4.2: The multi-graphs $(G_{V_8})_3$ and $(G_{V_8})_4$ are isomorphic; whereas $(G_{V_8})_3$ and $(G_{V_8})_5$ are non-isomorphic. In this way, the subsets $(V_8)_3$ and $(V_8)_4$ are isomorphic; whereas $(V_8)_3$ and $(V_8)_5$ are non-isomorphic. Algorithm 4.1.7 shows that there exist twenty-seven pattern subsets with eight points.

Below, we show the generalization of Algorithm 3.1.1 in Chapter 3. More concretely, Algorithm 4.1.7 is obtained by: (a) using as input data in Algorithm 3.1.1 the ordered set of vertices of the unit 4–cube, instead of that of the unit cube; and (b) changing the range of the first "for" loop.

Algorithm 4.1.7

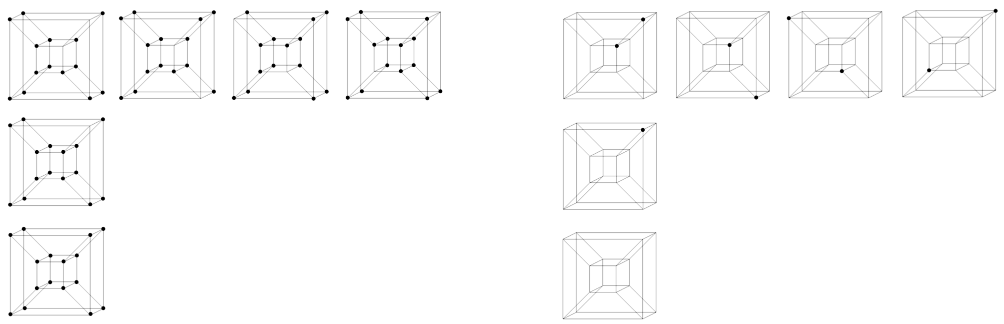
```
Input: $(V, \prec)$ set of vertices of the unit 4–cube together with an order relation $\prec$.
Output: pattern subsets with $8 \leq c \leq 16$ points.
begin
// $PS$: empty list to store the vertices of the non-isomorphic multi-graphs.
  for $c = 8, ..., 16$ do
    Construct an ordered set $(V_c, \prec)$ containing the $C(16, c)$ subsets with $c$ points
    {Lines 3–11 in Algorithm 3.1.1 in Chapter 3.}
      for each $(V_c)_i \subset V_c$ do
        Determine the multi-graph $(G_{V_c})_i$ associated with $(V_c)_i$
      end for
      for all $(V_c)_{i_1} \subset V_c$ and $(V_c)_{i_2} \subset V_c$ such that $(V_c)_{i_1} \prec (V_c)_{i_2}$ do
        if $(G_{V_c})_{i_1}$ and $(G_{V_c})_{i_2}$ are isomorphic then {$(V_c)_{i_1}$ and $(V_c)_{i_2}$ are isometric.}
          $V_c = V_c - \{(V_c)_{i_2}\}$
        end if
      end for
      $PS = PS \bigcup V_c$
  end for
  return $PS$
```

end

Remark 4.1.5. *Let us note that Algorithm 4.1.7 can only be used to construct the pattern subsets with at least 8 points, as it is shown in Remark 4.1.3.*

Remark 4.1.6. *Given an order relation $\prec$ on the set of vertices of the unit 4–cube, Algorithm 4.1.7 determines the smallest pattern subsets with respect to $\prec$. Moreover, by changing the order relation, other pattern subsets (isometric to these ones) are obtained.*

In Table 4.3, we show the pattern subsets in $\mathbb{Z}^4$ with $14 \leq c \leq 16$ points obtained by using lexicographic order in Algorithm 4.1.7, and we also show the subsets with $0 \leq c \leq 2$ points obtained by complementation.



Tab. 4.3: On the left: pattern subsets in $\mathbb{Z}^4$ with $14 \leq c \leq 16$ points obtained by using lexicographic order in Algorithm 4.1.7. On the right: pattern subsets in $\mathbb{Z}^4$ with $0 \leq c \leq 2$ points obtained by complementation.

The results shown in Theorem 4.1.7 (analogous to Theorem 3.1.6 in Chapter 3) are determined by Algorithm 4.1.7.

Theorem 4.1.7. *In $\mathbb{Z}^4$, there exist: (a) seventy-four pattern subsets with eight points; (b) fifty-six pattern subsets with nine points; (c) fifty pattern subsets with ten points; (d) twenty-seven pattern subsets with eleven points; (e) nineteen pattern subsets with twelve points; (f) six pattern subsets with thirteen points; (g) four pattern subsets with fourteen points; (h) one pattern subset with fifteen points; and (i) one pattern subset with sixteen points.*

Taking into account the complementation, we can formulate Corollary 4.1.8.

Corollary 4.1.8. *In $\mathbb{Z}^4$, there exist: (b') fifty-six pattern subsets with seven points; (c') fifty pattern subsets with six points; (d') twenty-seven pattern subsets with five points; (e') nineteen pattern subsets with four points; (f')* *six pattern subsets with three points; (g') four pattern subsets with two points; (h') one pattern subset with one point; and (i') one pattern subset with zero points.*

Remark 4.1.9. *The results shown in Theorem 4.1.7 and Corollary 4.1.8 confirm Pólya's count in 1940 (see Table II in [26]), whose main difficulty to count the different 2–colorings of the 4–cube was the derivation of the appropriate cycle indices (see [4] for more details).*

4.1.1.2 A second solution

In an analogous way as Chapter 3, a more efficient algorithm than Algorithm 4.1.7 has been defined in order to find the pattern subsets. Similarly, this algorithm applies the group of isometries of a 4–cube to the subsets of points. In this way, it returns the pattern subsets.

Computing the group of isometries of a 4–cube

Remark 4.1.10. *Analogously to Chapter 3,* ***Automorphisms[Hypercube[4]]*** *in the* Mathematica *package* ***Combinatorica'*** *gives the list with the 384 permutations of vertices which leave the unit 4–cube invariant. Each of these permutations corresponds to an isometry of the unit 4–cube.*

Remark 4.1.11. *Similarly, a standard implementation for computing the permutations of vertices which leave the unit 4–cube HC invariant can be conceived as follows.*

Let $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}, v_{12}, v_{13}, v_{14}, v_{15}, v_{16}\}$ *be the set of vertices of HC: (a)* v_1 *can be placed on sixteen different positions; (b) if* v_2 *is the end-point of an edge incident to* v_1*, then* v_2 *can be placed on four different positions since there exist four edges incident to* v_1*; (c) if* v_3 *is the end-point of another edge incident to* v_1*, then (once* v_2 *is placed)* v_3 *can be placed on three different positions; (d) if* v_4 *is the end-point of another edge incident to* v_1*, then (once* v_2, v_3 *are placed)* v_4 *can be placed on two different positions; (e) if* v_5 *is the end-point of another edge incident to* v_1*, then (once* v_2, v_3, v_4 *are placed)* v_5 *can be placed on only one position; analogously, (f) the positions of the vertices* $v_6, v_7, v_8, v_9, v_{10}, v_{11}, v_{12}, v_{13}, v_{14}, v_{15}, v_{16}$ *are fixed.*

In this way, we can conclude that there exist $16 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 384$ *elements in the group of isometries of HC.*

Applying the group of isometries to the subsets

Once computed the group *iso_4cube* of isometries of a 4–cube, we define Algorithm 4.1.8 for obtaining the pattern subsets from this group.

Algorithm 4.1.8

```
Input: $(V, \prec)$ set of vertices of the unit 4–cube together with an order relation $\prec$.
       iso_4cube: group of isometries of the unit 4–cube.
Output: pattern subsets with $0 \leq c \leq 16$ points.
begin
// $PS$: empty list to store the pattern subsets with $0 \leq c \leq 16$ points.
  for $c = 0, ..., 16$ do
    Construct an ordered set $(V_c, \prec)$ containing the $C(16, c)$ subsets with $c$ points
    {Lines 3–8 in Algorithm 3.1.2 in Chapter 3}
    for each $(V_c)_i \subset V_c$ ordered by $\prec$ do
      for each $\sigma \in iso\_cube - \{identity\}$ do
        $V_c = V_c - \{\sigma((V_c)_i)\}$
      end for
    end for
    $PS = PS \bigcup V_c$
  end for
  return $PS$
end
```

First, Algorithm 4.1.8 constructs an ordered set $(V_c, \prec)$ containing the $C(16, c)$ subsets with $0 \leq c \leq 16$ vertices of the unit 4–cube. Next, for $(V_c)_i \subset V_c$, it computes and removes any subset $(V_c)_j \neq (V_c)_i$ isometric to $(V_c)_i$. In this way, Algorithm 4.1.8 allows us to obtain a representative subset of each isometry class.

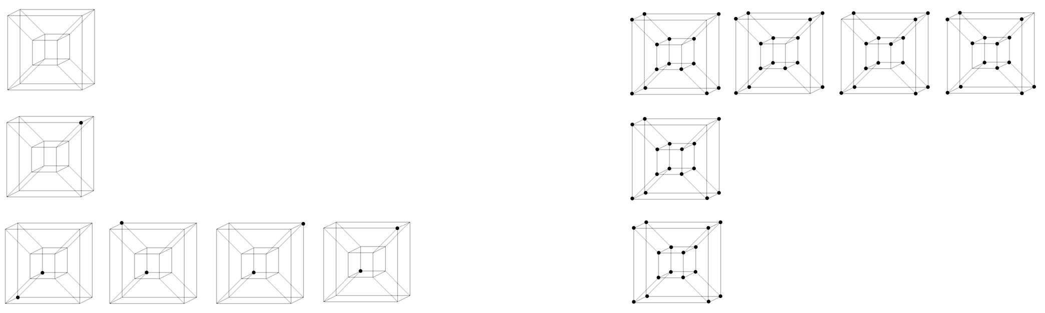
Remark 4.1.12. *Algorithm 4.1.8 not only improves the computational time of Algorithm 4.1.7 (see Table 4.4) but it can be used to construct pattern subsets regardless of the number of points of these.*

	Algorithm 4.1.7	**Algorithm 4.1.8**
***c* = 8**	11092.3 sec.	223.39 sec.
***c* = 9**	9482.27 sec.	189.3 sec.
***c* = 10**	11652.5 sec.	186.73 sec.
***c* = 11**	8382.22 sec.	112.68 sec.
***c* = 12**	2384.01 sec.	84.7 sec.
***c* = 13**	734.34 sec.	29.85 sec.
***c* = 14**	175.04 sec.	21.26 sec.
***c* = 15**	25.78 sec.	6.02 sec.

Tab. 4.4: Algorithm 4.1.7 constructs and checks $C(16,8) = 12870$, $C(16,9) = 11440$, $C(16,10) = 8008$, $C(16,11) = 4368$, $C(16,12) = 1820$, $C(16,13) = 560$, $C(16,14) = 120$ and $C(16,15) = 16$ multi-graphs in 11092.3, 9482.27, 11652.5, 8382.22, 2384.01, 734.34, 175.04 and 25.78 seconds of CPU time, respectively. These results are improved by Algorithm 4.1.8 to 223.39, 189.3, 186.73, 112.68, 84.7, 29.85, 21.26 and 6.02 seconds of CPU time, respectively. The computations have been made by using Mathematica 7.0.0 in AMD Turion(tm) 64 X2 Mobile Technology TL-58 1.90 GHz.

Remark 4.1.13. *Analogously to Algorithm 4.1.7, Algorithm 4.1.8 determines the smallest pattern subsets with respect to the order relation given on the set of vertices of the unit 4–cube; so by changing the order relation, other pattern subsets (isometric to those shown in Appendix A) are obtained.*

In Table 4.5, we show the pattern subsets in $\mathbb{Z}^4$ with 0,1,2,14,15,16 points obtained by using lexicographic order in Algorithm 4.1.8.



Tab. 4.5: Pattern subsets in $\mathbb{Z}^4$ with 0,1,2,14,15,16 points obtained by using lexicographic order in Algorithm 4.1.8. The pattern subsets on the left coincide (up to isometries) with these on the right in Table 4.3. The pattern subsets on the right coincide with these on the left in Table 4.3.

Remark 4.1.14. *Theorem 4.1.7 and Corollary 4.1.8 can be proved by using Algorithm 4.1.8.*

Following the structure of Chapter 3, below, we associate each of the 402 pattern subsets with a pattern cell.

4.1.2 Pattern cells

We generalize the computational method shown in Section 3.1.2 in Chapter 3 in order to determine the convex hull of the pattern subsets in $\mathbb{Z}^4$. In a similar way, the technique consists in deforming the 4–cube which contains to the subset. The deformation is a consequence of degenerating the edges of the 4–cube incident to white vertices. In this case, the degeneracies of the edges of the 4–cube can lead to the deformations and/or degeneracies of faces, volumes and hypervolumes contained in the 4–cube (see Figure 4.4).

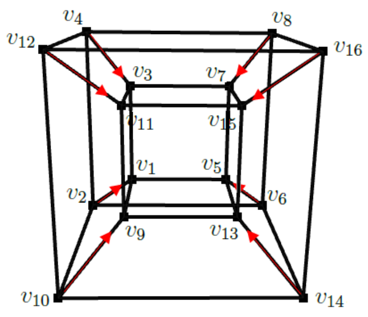


Fig. 4.4: The degeneracies of the edges $v_1v_2, v_3v_4, v_5v_6, v_7v_8, v_9v_{10}, v_{11}v_{12}, v_{13}v_{14}, v_{15}v_{16}$ into the vertices $v_2, v_4, v_6, v_8, v_{10}, v_{12}, v_{14}, v_{16}$, respectively, lead to the fact that the 4–cube is degenerated into the cube $v_1v_3v_5v_7v_9v_{11}v_{13}v_{15}$.

Remark 4.1.15. *Let us recall that in dimension 3, the convex hull may not be obtained by degenerating edges not belonging to the cube. In dimension 4, this problem is inherited. It suffices to embed in the 4–cube the example shown in Figure 3.15 in Chapter 3. Nevertheless, in Figure 4.5, we show a 4–dimensional example obtained by extending a similar case to that shown in Figure 3.15 in Chapter 3.*

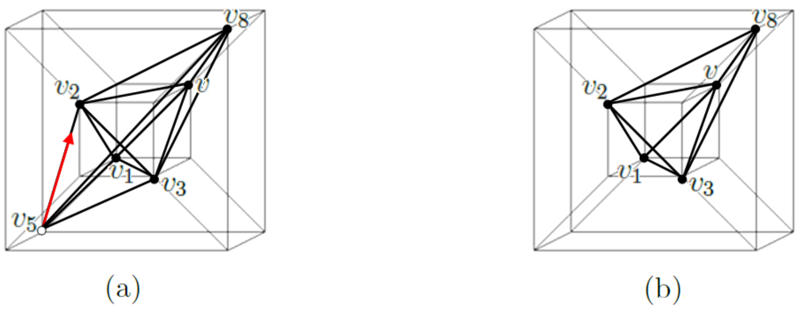


Fig. 4.5: (a) The hypervolume with six vertices is deformed by degenerating the edge v_2v_5. (b) The vertices v, v_1, v_2, v_3, v_8 define a hypertetrahedron, but the convex hull of these vertices cannot be obtained from (a). Moreover, we obtain similar results by degenerating the edge vv_5 or v_3v_5, but if we degenerate the edge v_1v_5 or v_5v_8, we obtain the convex hull.

Remark 4.1.16. *Analogously to dimension 3, the edges of the 4–cube have to degenerate following a certain order; otherwise, at one of the steps of the deformation of the 4–cube, the white vertex to treat may not be incident to any edge of the 4–cube. Figure 4.6 shows the generalization of the example presented in Figure 3.23 in Chapter 3.*

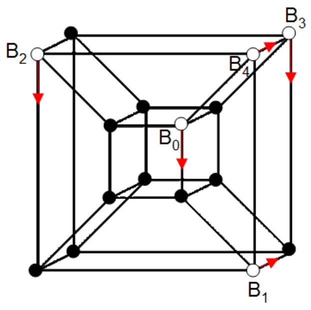


Fig. 4.6: If we degenerate (1) the edge incident to B_0 marked with an arrow, (2) the edge incident to B_1 marked with an arrow, (3) the edge incident to B_2 marked with an arrow, and (4) the edge incident to B_3 marked with an arrow, then there is not any edge of the 4–cube incident to B_4 for degenerating.

A procedure similar to that shown in Section 3.1.2.2 in Chapter 3 allows us to determine an order relation on the edge degeneracies, in such a way that at each step of the deformation of the 4–cube, there exists an edge of it incident to the white vertex to treat. More concretely, the procedure consists in: (1) considering the white vertices of the 4–cube and the edges which join them as a subgraph S of the 4–cube; and (2) constructing a rooted spanning *tree of each connected component of S, whose root is a white vertex adjacent to a black vertex of the 4–cube. These trees establish a hierarchy on the set of white vertices, and consequently, they determine an order for degenerating the edges incident to white vertices, where the last edge in degenerating is that made up by the root and by a black vertex adjacent to it. Let us note that in dimension 4, a vertex of the rooted spanning tree can be adjacent to at most four vertices of the tree, i.e. a father can have at most three children. See Figure 4.7 for an example.*

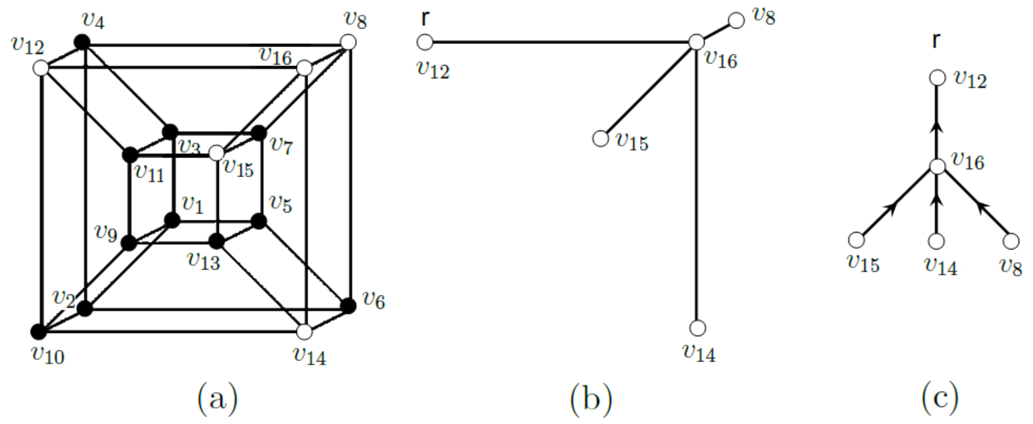


Fig. 4.7: From left to right: (a) one of the pattern subsets with eleven points; (b) the subgraph of the 4–cube obtained by deleting the black vertices (and, obviously, the edges incident to these vertices); (c) the rooted spanning tree of the only connected component of the subgraph. The root of the tree is denoted by r and the arrows indicate how to degenerate the edges.

4.1.2.1 *Convex hull of a pattern subset*

We extend the procedure shown in Chapter 3 in order to construct the convex hull of any pattern subset in $\mathbb{Z}^4$. In this way, we determine (up to isometries) the cells (together with their boundary) which can be obtained by deforming a 4–cube. Analogously, we degenerate the edges of the 4–cube incident to white vertices.

As we commented at the beginning of Section 4.1.2, the degeneracies of edges of the 4–cube incident to white vertices can lead to the degeneracies of faces, volumes, and/or hypervolumes contained in the 4–cube. Below, we relate the existence of degenerated faces, volumes, and/or hypervolumes to the star of the degenerated edge and to that of the white vertex to treat.

Let $(V_c)_i$ be a pattern subset contained in the 4–cube $HC_{(V_c)_i}$, and let S be the subgraph of $HC_{(V_c)_i}$ constructed by deleting the points of $(V_c)_i$. For each white vertex B of $HC_{(V_c)_i}$, we consider the rooted spanning tree T of the connected component of S containing B, and the edge $e = AB$ of T whose end-points are B and its father. Let $S(B)$ be the star of the white vertex to treat, and let $S(e)$ be the star of the edge e. Next, we degenerate e.

Cell deformations

– The faces contained in $S(B)$ are deformed under the same conditions as in the three-dimensional case.

– Let v be a volume contained in $S(B)$.

1. If $v \notin S(e)$ is a tetrahedron, then v is deformed into another tetrahedron v'. The vertices of v' are those of v except B which is replaced with the other end-point of the edge e. An example of this case is shown in Figure 4.8.

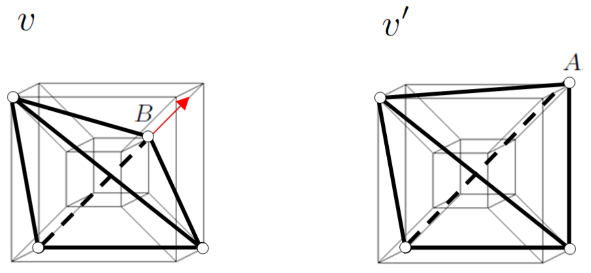


Fig. 4.8: The tetrahedron v is deformed into another tetrahedron v', as a consequence of degenerating an edge e satisfying that $v \notin S(e)$.

2. If $v \notin S(e)$ is a pyramid, then v is deformed into another (spatial or not) volume with five vertices. Below, we detail both cases:

   (a) If B is the apex of v, then v is deformed into another pyramid v' whose base is that of v. See Figure 4.9.

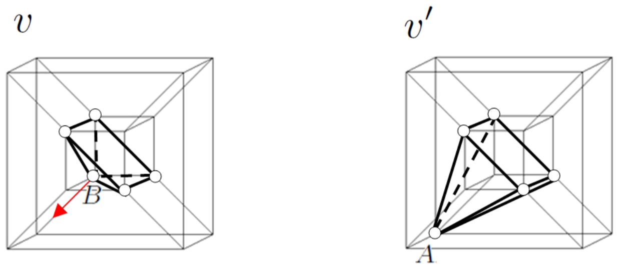


Fig. 4.9: A pyramid v is deformed into another pyramid v', as consequence of degenerating an edge incident to the apex of v.

(b) If B is a vertex of the base of v, then v is deformed into a volume v' of five non-cospatial vertices. In this case, in order to avoid convexity problems related to this non-spatial volume, we attach the triangular face whose vertices are the other end-points of the edges of v incident to B. This face allows us to cut up the non-spatial volume into two tetrahedra sharing a face. The shared face is the attached face. See Figure 4.10.

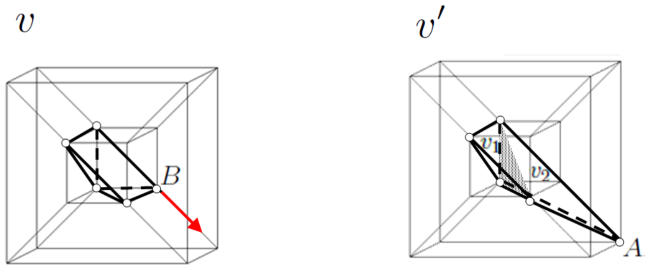


Fig. 4.10: A pyramid v is deformed into a non-spatial volume v' with five vertices, as a consequence of degenerating an edge e satisfying that $v \notin S(e)$. To avoid convexity problems, the shaded face is attached. This face cuts up the non-spatial volume into two tetrahedra v_1 and v_2.

3. If $v \notin S(e)$ is a volume with $p \geq 5$ vertices different from a pyramid, then v is deformed into a volume v' of $p \geq 5$ non-cospatial vertices. In this case, in order to avoid convexity problems related to this non-spatial volume, we attach the face f whose vertices are the other end-points of the edges of v incident to B. This face allows us to cut up the non-spatial volume v' into two volumes v_1 and v_2 (see Figure 4.11). More concretely, v_1 is the volume consisted of $p-1$ cospatial vertices which coincide with those of v except B; and v_2 is the volume whose vertices are the other end-point of e and the vertices of the attached face. Moreover, the relation between the vertices of f and those of v_2 is shown in Corollary 4.1.17.

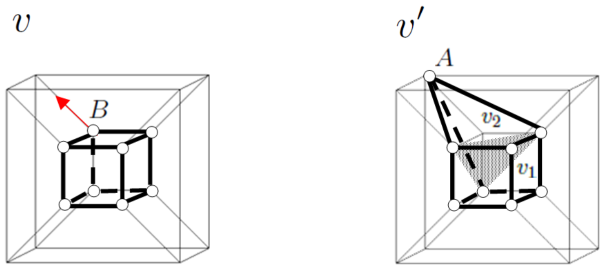


Fig. 4.11: The volume v' of non-cospatial vertices is cut up into two volumes with a common face.

If f is a face of four non-coplanar vertices, then, in order to avoid convexity problems related to this non-planar face, we attach the edge XX' which allows us to cut up the non-planar face into two triangular faces sharing an edge. The shared edge is the attached edge.

Moreover, Corollary 4.1.17 assures that v_2 is a volume of five non-cospatial vertices. In order to avoid convexity problems related to this non-spatial volume, we attach the face whose vertices are X, X' and the other end-point of e. This face allows us to cut up the non-spatial volume v_2 into two tetrahedra sharing a face. The shared face is the attached face. Furthermore, each of these tetrahedra is attached to v_1 by one of the triangular faces forming f. All this has allowed us to cut up the non-spatial volume v' into three volumes sharing a face two by two. See Figure 4.12.

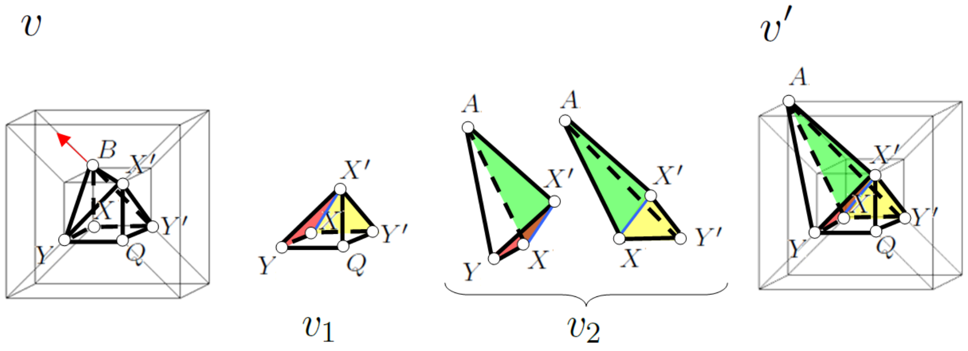


Fig. 4.12: The volume $v' = AXX'YY'Q$ of non-cospatial vertices is cut up into the volumes $v_1 = XX'YY'Q$ of cospatial vertices and $v_2 = AXX'YY'$ of non-cospatial vertices sharing the non-planar face $f = XX'YY'$. Moreover, f is cut up into two triangular faces sharing an edge, and v_2 is cut up into two tetrahedra $AXX'Y$ and $AXX'Y'$ sharing a face.

Corollary 4.1.17. *f is a face of four non-coplanar vertices if and only if v_2 is a volume of five non-cospatial vertices. Moreover, by replacing the other end-point of e with B in the set of vertices of v_2, we obtain (up to isometry) the pattern cell $C((V_5)_2)$ (see Table 3.6 in Chapter 3).*

Cell degeneracies

- The faces contained in $S(B)$ are degenerated under the same conditions as in the three-dimensional case.
- The volumes contained in $S(B)$ are degenerated under the same conditions as in the three-dimensional case.
- Let HV be a hypervolume contained in $HC_{(V_c)_i}$, and let $\mathcal{V}$ be the set of volumes of HV.
  If there exists only one volume $O \in \mathcal{V} - S(B)$, then HV degenerates into O. This degenerated hypervolume must be removed starting from a volume incident to it (see Figure 4.13). The degenerated edge leads to a degenerated hypervolume.

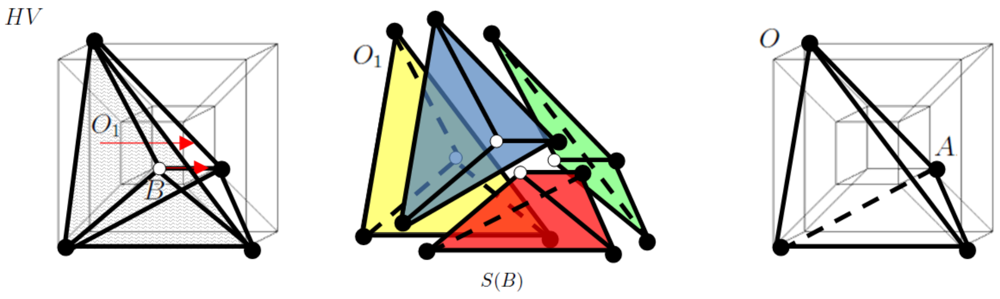


Fig. 4.13: There exists only one tetrahedron O of the hypertetrahedron HV which does not belong to $S(B)$, so HV degenerates into O. This degenerated hypertetrahedron is removed starting from the only volume O_1 in $S(B)-S(e)$.

The corresponding operations in the three-dimensional case led to the fact that two coplanar triangular faces sharing an edge may appear. In the four-dimensional case, cospatial volumes (with up to seven vertices) sharing a face may have appeared. By reasoning in a similar way as Chapter 3, we remove the shared face and the edges which are inside of another volume. More concretely, in dimension four, it may have appeared (a) three cospatial volumes with common faces two by two sharing an edge. In this case, we remove the shared edge, three common faces, and three volumes, and we attach a new volume made by the union of those three faces (see Figure

4.14); and it may also have appeared (b) two cospatial volumes sharing a face. In this case, we remove the common face, and both volumes, and we attach a new volume made up by the union of those two volumes (see Figure 4.15).

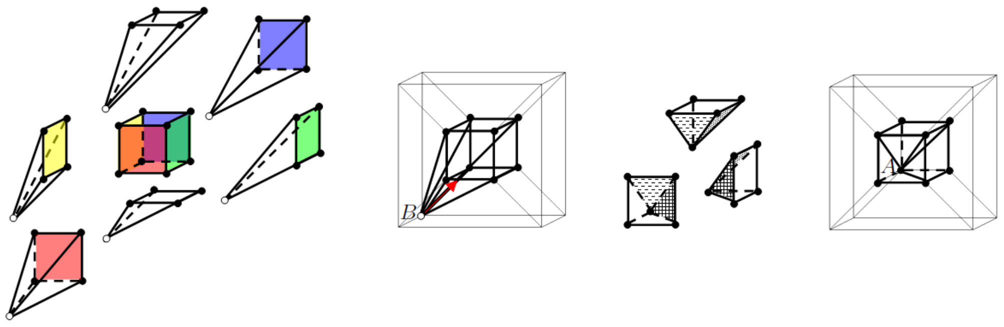


Fig. 4.14: By degenerating an edge of a hypervolume with nine vertices, which consists of a cube with one pyramid attached on each of its faces, we obtain three pyramids forming a cube with an edge inside of it.

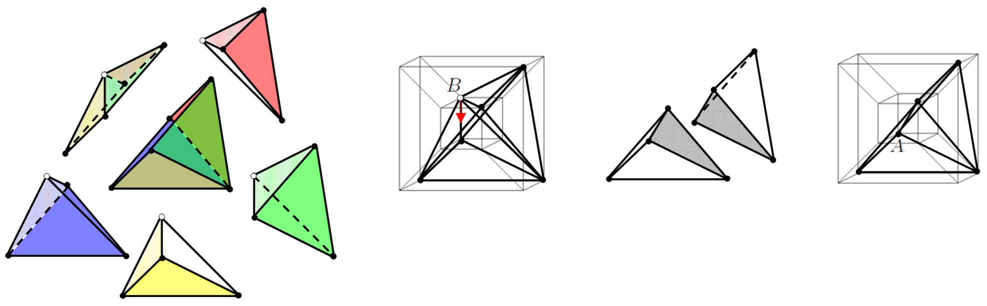


Fig. 4.15: By degenerating an edge of a hypervolume with six vertices, which consists of one pyramid attached to other on its square face, and four tetrahedra attached to the corresponding pairs of triangular faces of both pyramids, we obtain two cospatial tetrahedra sharing a face. By removing this face, we obtain a pyramid.

Summarizing, the technique developed for constructing the convex hull of a pattern subset in $\mathbb{Z}^4$ includes, in addition to the operations of the three-dimensional case, the steps of: (1) attaching edges and faces for converting non-spatial volumes into spatial volumes sharing a face two by two; (2) removing faces and edges for converting cospatial volumes into only one volume; and (3) studying the degenerated 4–cells.

The previous results allow us to define a procedure (Algorithm $4D$) for: (1) deforming the unit 4–cube into the convex hull of any pattern subset

$(V_c)_i \subset \mathbb{Z}^4$; (2) constructing the cell defined by this pattern subset; and (3) computing its boundary.

Remark 4.1.18. *Given a pattern subset $(V_c)_i$, Algorithm $4D$ deforms and degenerates the cells contained in the unit 4–cube HC_V for computing the convex hull of the points of $(V_c)_i$. In this way, the cell $C((V_c)_i)$ defined by $(V_c)_i$ is determined by the points inside of this convex hull. Moreover, the boundary of $C((V_c)_i)$ is computed in terms of the vertices, edges, faces and volumes of the convex hull. More concretely, $\partial(C((V_c)_i))$ is given by Ve', Ed', Fa', Vo'.*

Remark 4.1.19. *Let us observe that Algorithm $4D$ generalizes Algorithm 3.1.3 in Chapter 3. Moreover, the cases treated in Algorithm 3.1.3 are included in Algorithm $4D$.*

In the same way as the three-dimensional case, by using as input the 402 pattern subsets shown in Appendix A, Algorithm $4D$ returns the 402 pattern cells shown in Appendix B. Let us note that 347 of them are 4–cells (see second column of Table 3 in [1]).

4.2 Construction of the cell complex

The same procedure used in the corresponding section in Chapter 3 allows us to construct a cell complex from a given 4–dimensional binary digital image. In the same way, the work is done in two levels: (a) the vertex level, where we compute the subsets of points of the image and we associate them with pattern subsets; and (b) the cell level, where we determine the cells of the complex from the pattern cells. Finally, the cell complex is constructed by attaching only one copy of each cell.

4.2.1 Vertex level

We *localize* and *classify* the subsets of black points of the image in each 4–cube of the grid. The *localization* is made by using an algorithm identical to Algorithm 3.2.4 in Chapter 3, where the grid is made up by 4–cubes. The *classification* of the subsets is done with an algorithm similar to Algorithm 3.2.5 in Chapter 3, where the group of isometries and the pattern subsets now correspond to those of the 4–cube.

In Example 4.2.1, we show the results of localizing and classifying the subsets of black points of a simple image. These results have been obtained by using algorithms close to Algorithm 3.2.4 and 3.2.5 in Chapter 3.

Example 4.2.1. *Let* $\{\{0,0,0,0\},\{0,1,0,0\},\{0,1,0,1\},\{0,1,1,0\},\{0,2,0,0\},$

$\{1,1,0,0\}\}$ *be the set of the 6 black points of a dual grid G made up by $2 \times 2 \times 2 \times 2$ 4–cubes.*

The scanning of the 16 4–cubes of the grid returns: (a) the set $V(I)$ with the 8 non-empty subsets of points of the image, and (b) the set $V(I)'$ with the 8 non-empty pattern subsets associated with each of the subsets of $V(I)$.
$V(I) = \{\{\{0,0,0,0\},\{0,1,0,0\},\{0,1,0,1\},\{0,1,1,0\},\{1,1,0,0\}\},\{\{0,1,0,1\}\},$
$\{\{0,1,1,0\}\},\{\{0,1,0,0\},\{0,1,0,1\},\{0,1,1,0\},\{0,2,0,0\},\{1,1,0,0\}\},\{\{0,1,0,1\}\},$
$\{\{0,1,1,0\}\},\{\{1,1,0,0\}\},\{\{1,1,0,0\}\}\}$
$V(I)' = \{\{\{0,0,0,0\},\{0,0,0,1\},\{0,0,1,0\},\{0,1,0,0\},\{1,0,0,0\}\},\{\{0,0,0,0\}\},$
$\{\{0,0,0,0\}\},\{\{0,0,0,0\},\{0,0,0,1\},\{0,0,1,0\},\{0,1,0,0\},\{1,0,0,0\}\},\{\{0,0,0,0\}\},$
$\{\{0,0,0,0\}\},\{\{0,0,0,0\}\},\{\{0,0,0,0\}\}\}$

The results shown in Theorem 4.2.2 (extension to dimension 4 of Theorem 3.2.4) are determined by algorithms closed to Algorithm 3.2.4 and 3.2.5 in Chapter 3.

Theorem 4.2.2. *Any 4–dimensional binary digital image on a dual grid is made up by combining (up to isometries) the 402 pattern subsets shown in Appendix A, taking into account that it is not necessary to use all them and any of them can be used more than once.*

4.2.2 Cell level

We *identify* the pattern cells of the cell complex and we *invert* the isometry from each subset of points of the image to its associated pattern subset. We *identify* each pattern subset of the image with a pattern cell according to Appendix B. As we have commented in the corresponding subsection in Chapter 3, by inverting the isometry between a subset X of points of the image and its associated pattern subset Y, we can determine the cell $C(X)$ from the pattern cell $C(Y)$.

Remark 4.2.3. *By applying this procedure to each subset of points of the image, we obtain the cells (together with their boundary) defined by each of these subsets. The cell complex is constructed by attaching only one copy of each of these cells along their boundaries. In this way, we can prove Theorem 4.2.4 which is a 4–dimensional version of Theorem 3.2.7.*

Theorem 4.2.4. *The cell complex constructed from a given 4–dimensional binary digital image on a dual grid is made up by combining (up to isometries) the 402 pattern cells shown in Appendix B, taking into account that it is not necessary to use all them and any of them can be used more than once.*

In Example 4.2.5, we show the cells obtained by inverting the isometries which associate the subsets of points of the image shown in Example 4.2.1 with the pattern subsets.

Example 4.2.5. *The six singleton subsets of the image shown in Example 4.2.1 correspond to 0–cells.*

$\{\{\{0,0,0,0\}\},\{\{0,1,0,0\}\},\{\{0,1,0,1\}\},\{\{0,1,1,0\}\},\{\{0,2,0,0\}\},\{\{1,1,0,0\}\}\}$

The fourteen subsets made up by two points correspond to 1–cells.

$\{\{\{0,0,0,0\},\{0,1,0,1\}\},\{\{0,0,0,0\},\{0,1,1,0\}\},\{\{0,0,0,0\},\{1,1,0,0\}\},$
$\{\{0,1,0,0\},\{0,0,0,0\}\},\{\{0,1,0,0\},\{0,1,0,1\}\},\{\{0,1,0,0\},\{0,1,1,0\}\},$
$\{\{0,1,0,0\},\{0,2,0,0\}\},\{\{0,1,0,0\},\{1,1,0,0\}\},\{\{0,1,0,1\},\{0,1,1,0\}\},$
$\{\{0,1,0,1\},\{0,2,0,0\}\},\{\{0,1,0,1\},\{1,1,0,0\}\},\{\{0,1,1,0\},\{0,2,0,0\}\},$
$\{\{0,1,1,0\},\{1,1,0,0\}\},\{\{0,2,0,0\},\{1,1,0,0\}\}\}$

The sixteen subsets made up by three points correspond to 2–cells.

$\{\{\{0,0,0,0\},\{0,1,0,1\},\{0,1,1,0\}\},\{\{0,0,0,0\},\{0,1,0,1\},\{1,1,0,0\}\},$
$\{\{0,0,0,0\},\{0,1,1,0\},\{1,1,0,0\}\},\{\{0,1,0,0\},\{0,0,0,0\},\{0,1,0,1\}\},$
$\{\{0,1,0,0\},\{0,0,0,0\},\{0,1,1,0\}\},\{\{0,1,0,0\},\{0,0,0,0\},\{1,1,0,0\}\},$
$\{\{0,1,0,0\},\{0,1,0,1\},\{0,1,1,0\}\},\{\{0,1,0,0\},\{0,1,0,1\},\{0,2,0,0\}\},$
$\{\{0,1,0,0\},\{0,1,0,1\},\{1,1,0,0\}\},\{\{0,1,0,0\},\{0,1,1,0\},\{0,2,0,0\}\},$
$\{\{0,1,0,0\},\{0,1,1,0\},\{1,1,0,0\}\},\{\{0,1,0,0\},\{0,2,0,0\},\{1,1,0,0\}\},$
$\{\{0,1,0,1\},\{0,1,1,0\},\{0,2,0,0\}\},\{\{0,1,0,1\},\{0,1,1,0\},\{1,1,0,0\}\},$
$\{\{0,1,0,1\},\{0,2,0,0\},\{1,1,0,0\}\},\{\{0,1,1,0\},\{0,2,0,0\},\{1,1,0,0\}\}\}$

The nine subsets made up by four points correspond to 3–cells.

$\{\{\{0,0,0,0\},\{0,1,0,1\},\{0,1,1,0\},\{1,1,0,0\}\},\{\{0,1,0,0\},\{0,0,0,0\},$
$\{0,1,0,1\},\{0,1,1,0\}\},\{\{0,1,0,0\},\{0,0,0,0\},\{0,1,0,1\},\{1,1,0,0\}\},$
$\{\{0,1,0,0\},\{0,0,0,0\},\{0,1,1,0\},\{1,1,0,0\}\},\{\{0,1,0,0\},\{0,1,0,1\},$
$\{0,1,1,0\},\{0,2,0,0\}\},\{\{0,1,0,0\},\{0,1,0,1\},\{0,1,1,0\},\{1,1,0,0\}\},$
$\{\{0,1,0,0\},\{0,1,0,1\},\{0,2,0,0\},\{1,1,0,0\}\},\{\{0,1,0,0\},\{0,1,1,0\},$
$\{0,2,0,0\},\{1,1,0,0\}\},\{\{0,1,0,1\},\{0,1,1,0\},\{0,2,0,0\},\{1,1,0,0\}\}\}$

The two subsets made up by five points correspond to 4–cells.

$\{\{\{0,0,0,0\},\{0,1,0,0\},\{0,1,0,1\},\{0,1,1,0\},\{1,1,0,0\}\},$
$\{\{0,1,0,0\},\{0,1,0,1\},\{0,1,1,0\},\{0,2,0,0\},\{1,1,0,0\}\}\}$

4.3 Simplification of the cell complex

We conceive Algorithm 4.3.9 which extends Algorithm 3.3.6 defined in Chapter 3 to dimension four. In this way, Algorithm 4.3.9 simplifies the cell complex constructed from a given 4–dimensional binary digital image. More concretely, the *simplified complex* is given by: (a) the 0,1,2,3–cells of the initial complex which are not incident to a 4–cell; and (b) the set of 3–cells (together with their boundary) incident to exactly one of the 4–cells of the initial complex.

Algorithm 4.3.9

Input: $C_i(V(I))$: i–cells of the cell complex constructed from the image I, for $i = 0, 1, 2, 3, 4$.
$\partial(C_4(V(I)))$: boundary of the cells in $C_4(V(I))$.
Output: simplified cell complex.

```
begin
// Bord: empty list to store the cells of the simplified complex.
  for $i \in \{0,1,2,3\}$ do
    for each $i$–cell $c \in C_i(V(I))$ do
      if $c$ is not incident to a 4–cell then
        $Bord = Bord \bigcup \{c\}$
      else
        if $i \neq 3$ or $c$ is incident to exactly one 4–cell then
          $Bord = Bord \bigcup \{c\} \bigcup \{\partial c\}$
        end if
      end if
    end for
  end for
  return $Bord$
end
```

Example 4.3.1. *Algorithm 4.3.9 is applied to the cell complex whose cells are shown in Example 4.2.5. This algorithm determines that all the 0,1,2,3–cells are incident to a 4–cell, and it computes the eight 3–cells incident to exactly one of the 4–cells of the complex.*
$Bord = \{\{\{0,0,0,0\},\{0,1,0,0\},\{0,1,0,1\},\{0,1,1,0\}\},\{\{0,0,0,0\},\{0,1,0,0\},$
$\{0,1,0,1\},\{1,1,0,0\}\},\{\{0,0,0,0\},\{0,1,0,0\},\{0,1,1,0\},\{1,1,0,0\}\},\{\{0,0,0,0\},$
$\{0,1,0,1\},\{0,1,1,0\},\{1,1,0,0\}\},\{\{0,1,0,0\},\{0,1,0,1\},\{0,1,1,0\},\{0,2,0,0\}\},$
$\{\{0,1,0,0\},\{0,1,0,1\},\{0,2,0,0\},\{1,1,0,0\}\},\{\{0,1,0,0\},\{0,1,1,0\},$
$\{0,2,0,0\},\{1,1,0,0\}\},\{\{0,1,0,1\},\{0,1,1,0\},\{0,2,0,0\},\{1,1,0,0\}\}\}$

5. CONCLUSIONS AND FUTURE WORKS

In this thesis we have developed a procedure for extracting cell complexes from dual binary digital images of dimension 3 and 4.

The first step of the procedure consists in determining the non-isometric subsets of points which can be obtained from the vertices of a cube (resp. 4–cube), and constructing the convex hull of these subsets of points. The second step is scanning each subset of vertices of a cube (resp. 4–cube) of the grid and associating it with one of the non-isometric subset determined in the first step. Finally, we consider the convex hull of the non-isometric subset and we invert the isometry between this one and its corresponding subset of vertices, in this way, we obtain the cells of the cell complex.

The non-isometric subsets of points are determined by using a standard algorithm whose input are (1) all the subsets of points which can be determined from the vertices of a cube (resp. 4–cube); and (2) the group of isometries of a cube (resp. 4–cube).

The convex hull of the non-isometric subsets is constructed into two steps: (1) a first step of "deformation" and "degeneracy" of cells of a cube (resp. 4–cube); and (b) a second step of "correction" which allows us to assure the convexity.

Let us observe that by constructing the convex hull we work with the maximal neighborhood, i.e. with the 26–neighborhood (resp. 80–neighborhood). The results of working with other types of neighborhoods, for example the 6,18–neighborhood (resp. 8,32,64–neighborhood), can be obtained by removing the edges of the convex hull which are not defined in these types of neighborhoods.

The procedure developed in this thesis can be generalized to higher dimension. Moreover, (1) for determining the non-isometric subsets in nD it suffices to compute the group of isometries of an n–cube; and (2) for constructing the convex hull of these subsets it suffices to use the deformations and degeneracies of the cells of an $(n-1)$–cube, and to study the deformations and degeneracies of the cells of dimension $n-1$ of an n–cube.

In a near future is our interest to develop a similar procedure for extracting cell complexes from dual binary digital images of any dimension and by using any type of neighborhood.

BIBLIOGRAPHY

[1] O. Aichholzer. Extremal properties of 0/1-polytopes of dimension 5. In G. Ziegler and G. Kalai (eds.), Polytopes - Combinatorics and Computation, Birkhäuser, 111–130, 2000.

[2] R. C. Alperin. 2–Colorings of Cube Edges With 6 Each.

[3] J. Arias-Fisteus, N. Fernández-García, L. Sánchez-Fernández and C. Delgado-Kloos. Hashing and canonicalizing Notation 3 graphs. Journal of Computer and System Sciences 76(7): 663–685, 2010.

[4] D. C. Banks, S. A. Linton and P. K. Stockmeyer. Counting Cases in Substitope Algorithms. IEEE Transactions on Visualization and Computer Graphics 10(4): 371–384, 2004.

[5] B. Bollobas. Graph Theory: An Introductory Course. Graduate Texts in Mathematics, New York: Springer-Verlag, 1979.

[6] E. V. Chernyaev. Marching Cubes 33: Construction of Topologically Correct Isosurfaces. Technical Report CN/95-17, 1995.

[7] M. Couprie, F. N. Bezerra and G. Bertrand. Topological operators for grayscale image processing. Journal of Electronic Imaging 10(4): 1003–1015, 2001.

[8] M. Couprie and G. Bertrand. New characterizations of simple points, minimal non-simple sets and p-simple points in 2d, 3d and 4d discrete spaces. In D. Coeurjolly, I. Sivignon, L. Tougne and F. Dupont (eds.), Discrete Geometry for Computer Imagery, Berlin Heidelberg: Springer-Verlag 105–116, 2008.

[9] G. Damiand. Définition et étude d'un modèle topologique minimal de représentation d'images 2d et 3d. PhD thesis, Montpellier II University, 2001.

[10] G. Damiand. Topological model for 3D Image Representation: Definition and Incremental Extraction Algorithm. Computer Vision and Image Understanding 109(3): 260–289, 2008.

[11] S. E. Han. A generalized digital (k_0, k_1)-homeomorphism. Note di Matematica 22(2): 157–166, 2004.

[12] F. Hanisch. Marching square. CGEMS: Computer graphics educational materials source, 2008.

[13] R. Fontana. Regular Polyhedra and Symmetry. Math and Science in the World. Miscellaneous.

[14] Y. Kenmochi and A. Imiya. Combinatorial boundary of a 3D lattice point set. Visual Communication and Image Representation 17(4): 738–766, 2006.

[15] Y. Kenmochi, A. Imiya and N. F. Ezquerra. Polyhedra generation from lattice points. In S. Miguet, A. Montanvert and S. Ubéda (eds.), Discrete Geometry for Computer Imagery, Berlin Heidelberg: Springer-Verlag, 127–138, 1996.

[16] Y. Kenmochi, A. Imiya and A. Ichikawa. Boundary Extraction of Discrete Objects. Computer Vision and Image Understanding 71(3): 281–293, 1998.

[17] T. Y. Kong and A. W. Roscoe. A theory of binary digital pictures. Computer Vision, Graphics, and Image Processing 32(2): 221–243, 1985.

[18] W. G. Kropatsch. Building irregular pyramids by dual-graph contraction. Vision, Image and Signal Processing, 142(6): 366–374, 1995.

[19] L. J. Grady and J. R. Polimeni. Discrete Calculus: Applied Analysis on Graphs for Computational Science. Springer, 2010.

[20] J. L. Gross and J. Yellen. Handbook of Graph Theory. CRC Press, 2004.

[21] J.-O. Lachaud. Extraction de surfaces à partir d'images tridimensionnelles: approche discrète et approche par modèle déformable. PhD thesis, Joseph Fourier University, 1998.

[22] P. Lienhardt. N-dimensional generalized combinatorial maps and cellular quasi-manifolds. International Journal on Computational Geometry and Applications, 4(3): 275–324, 1994.

[23] W. E. Lorensen and H. E. Cline. Marching cubes: A high-resolution 3D surface construction algorithm. Computer Graphics 21(4): 163–169, 1987.

[24] B. K. Natarajan. On generating topologically consistent isosurfaces from uniform samples. The Visual Computer, 11(1): 52–62, 1994.

[25] T. S. Newman and H. Yi. A survey of the marching cubes algorithm. Computers & Graphics, 30(5): 854–879, 2006.

[26] G. Pólya. Sur Les Types Des Propositions Composées. The Journal of Symbolic Logic 5(3): 98–103, 1940.

[27] A. Rosenfeld. Adjacency in digital pictures. Information and Control 26(1): 24–33, 1974.

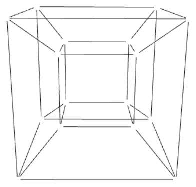
[28] A. V. Gelder and J. Wilhelms. Topological considerations in isosurface generation. ACM Transactions on Graphics, 13(4): 337–375, 1994.

[29] R. Meyssonnier. Régis Meyssonnier - France | LinkedIn[Proffesional Network]. Website: http://fr.linkedin.com/pub/r%C3%A9gis-meyssonnier/41/77b/866/en

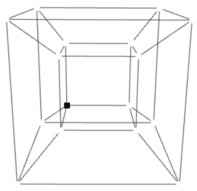
APPENDIX

A. PATTERN SUBSETS IN $\mathbb{Z}^4$

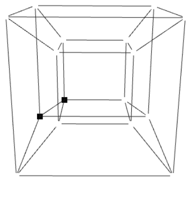
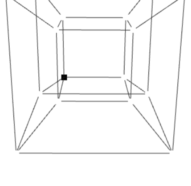
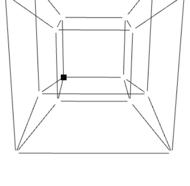
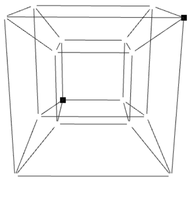
with 0 vertices



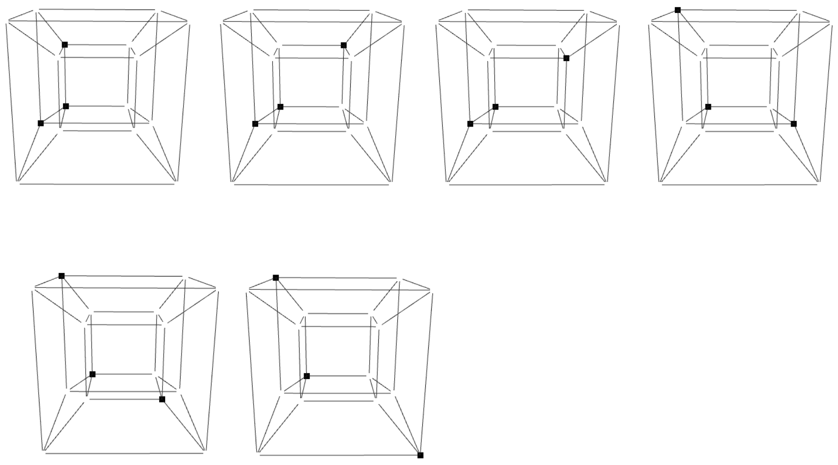
with 1 vertex



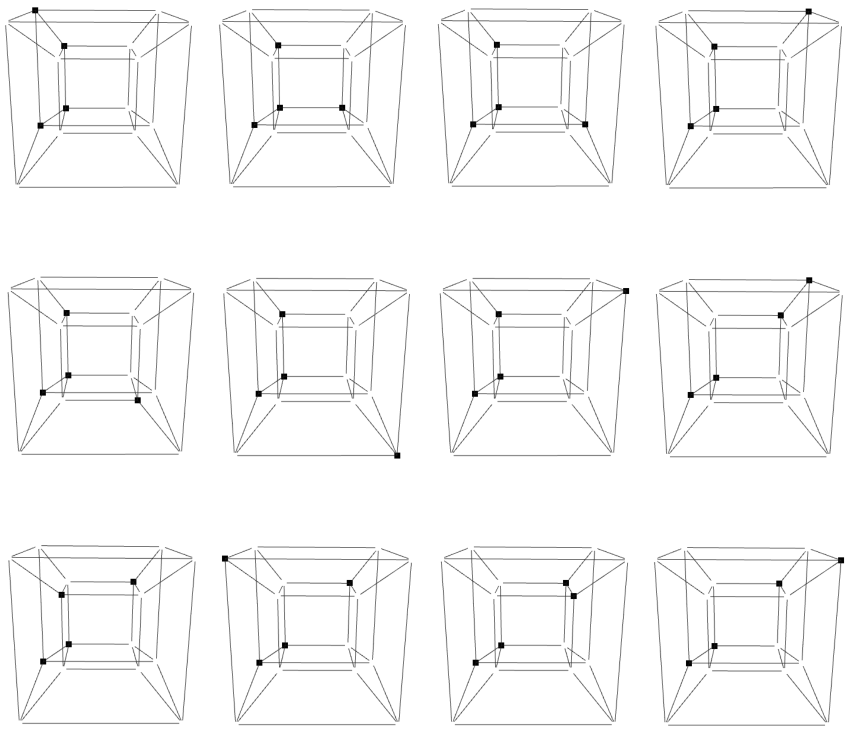
with 2 vertices

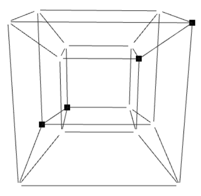
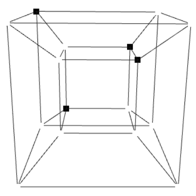
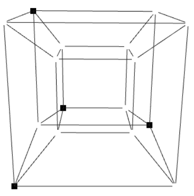
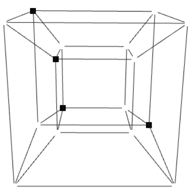





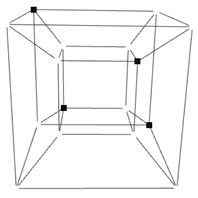
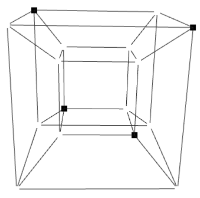
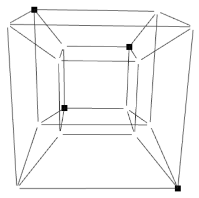
with 3 vertices



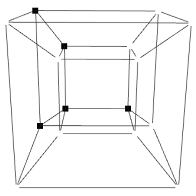
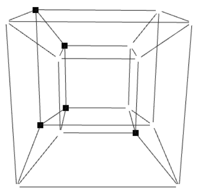
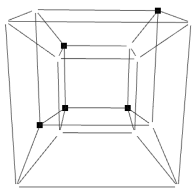
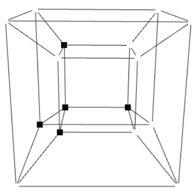
with 4 vertices

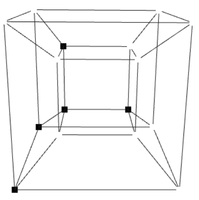
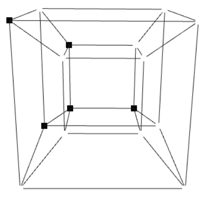
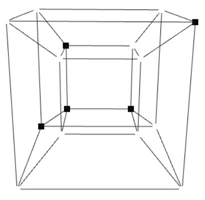
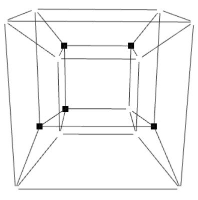


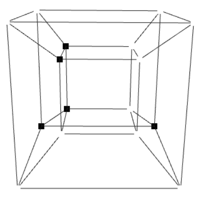
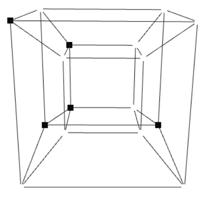
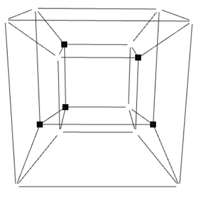
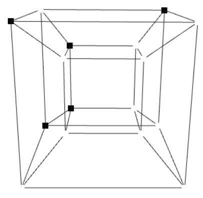





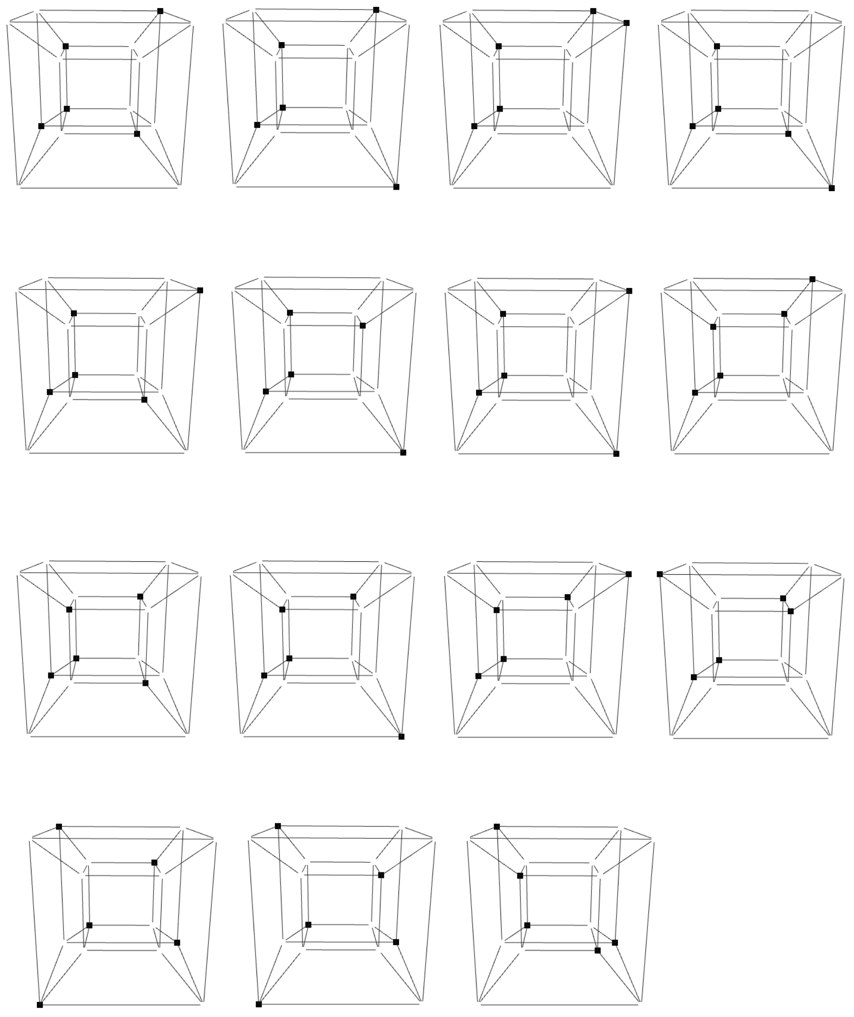




with 5 vertices

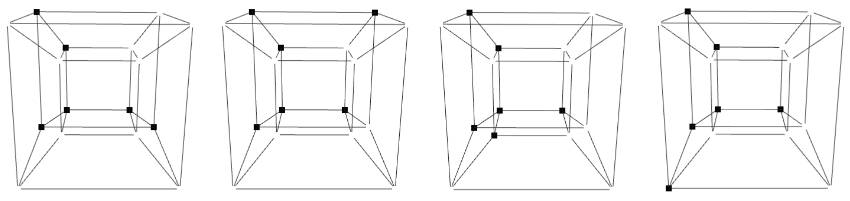





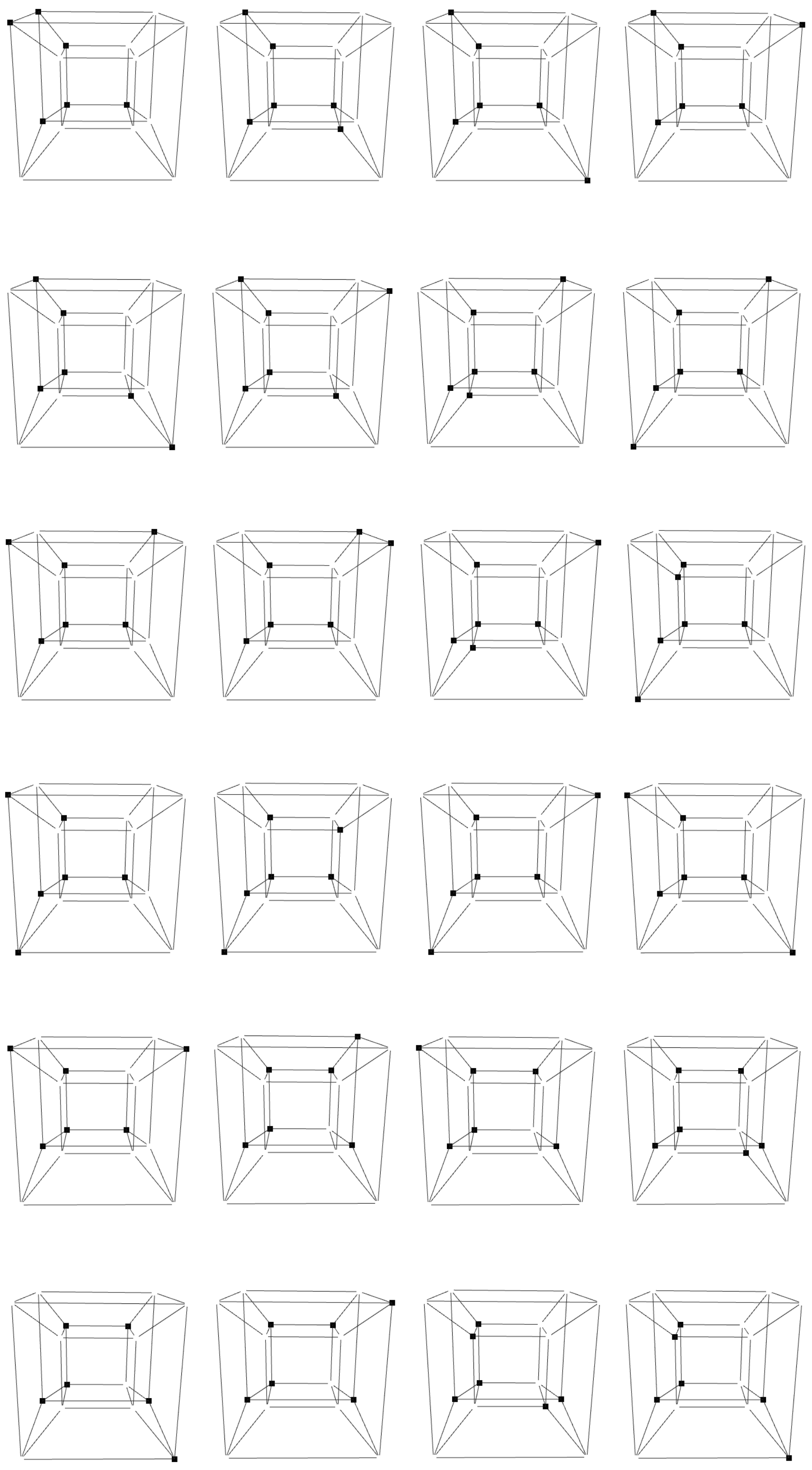





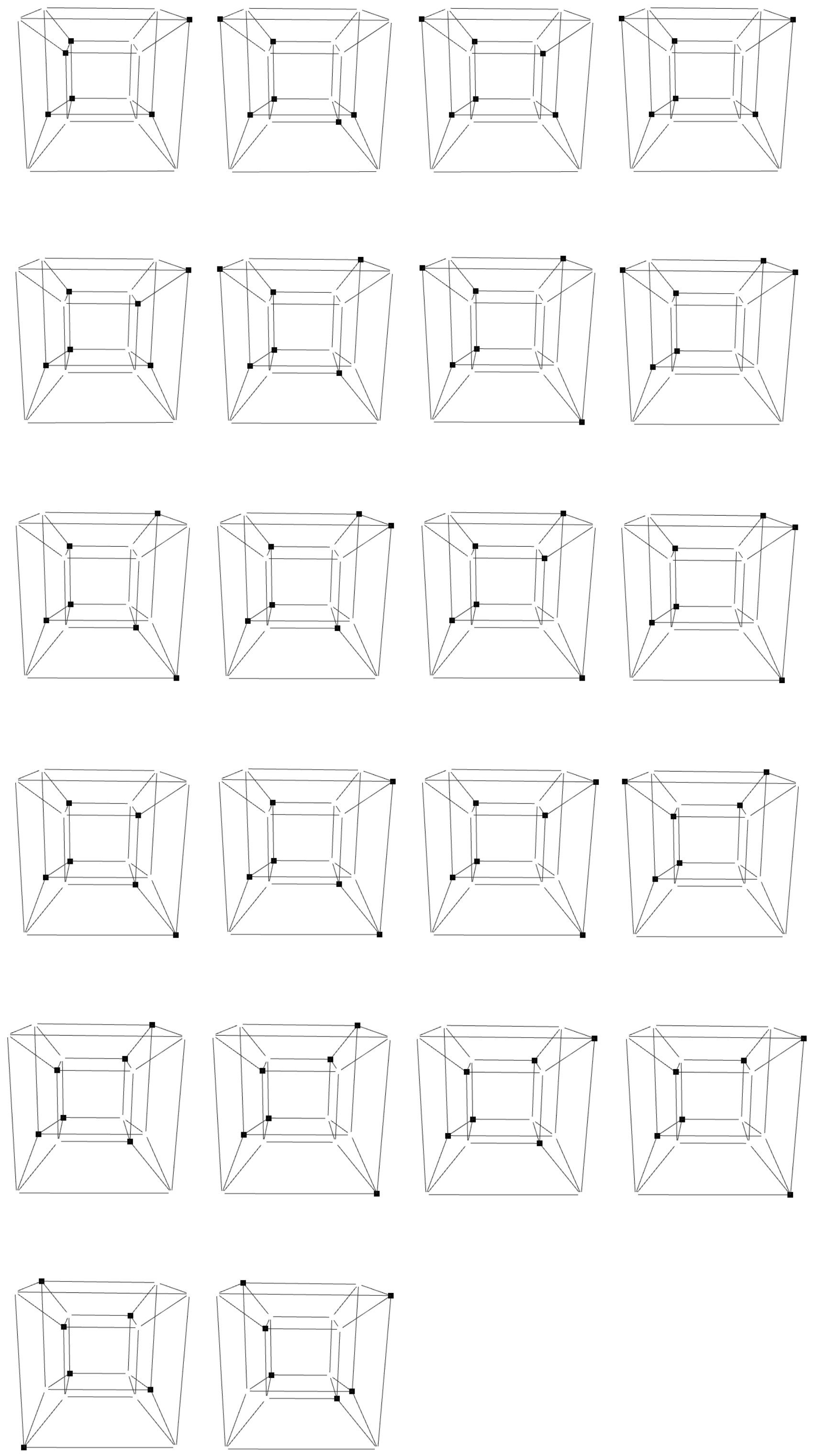







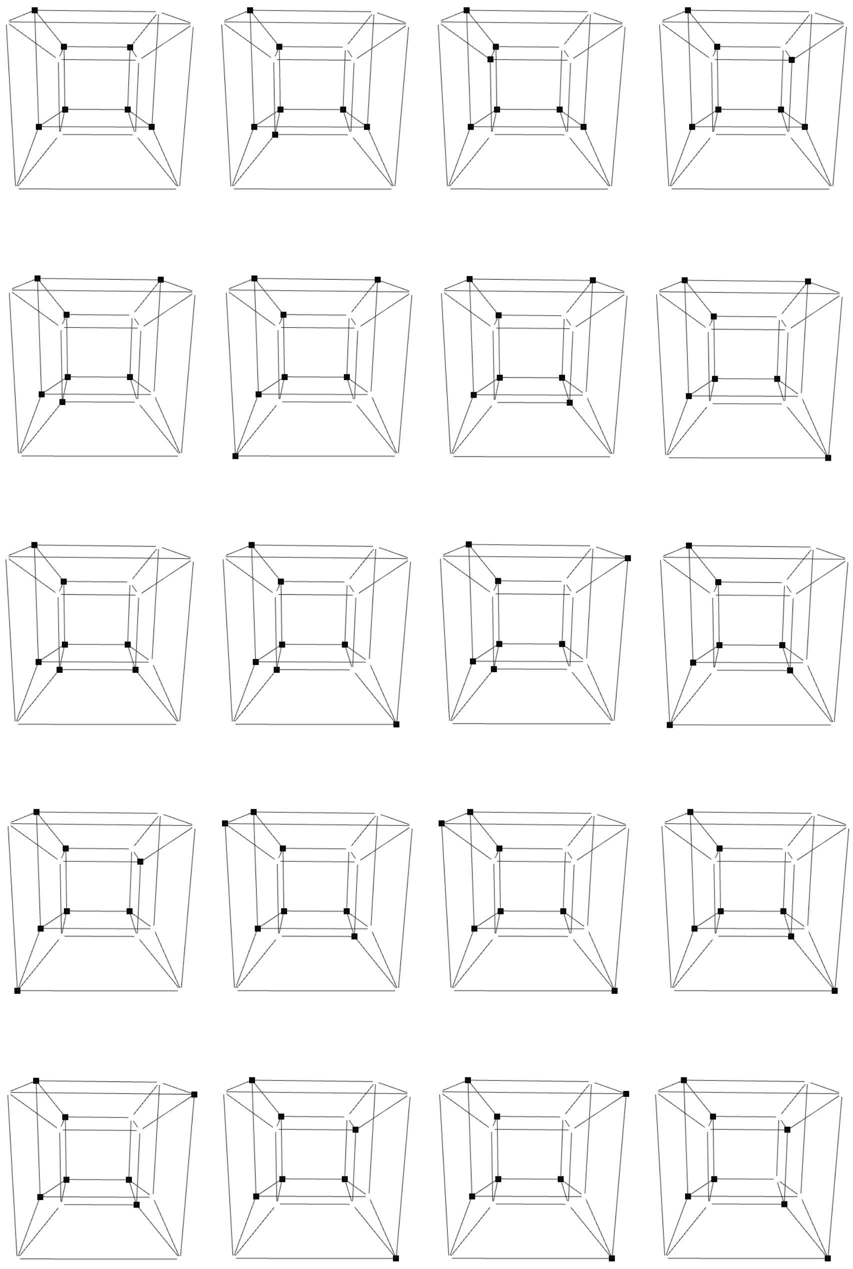
with 6 vertices

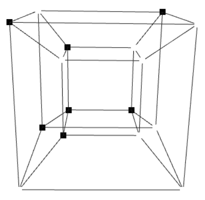
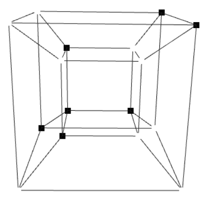
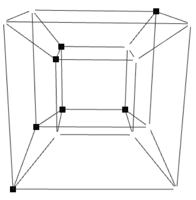
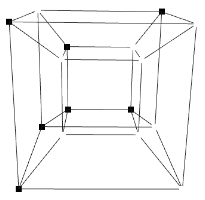
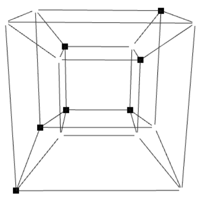
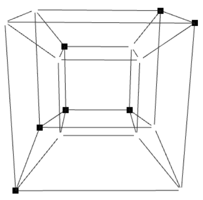
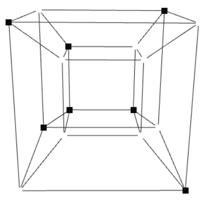
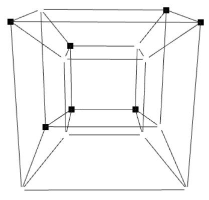
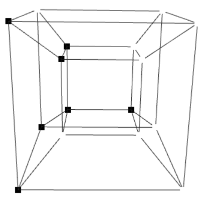
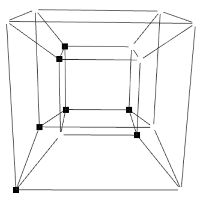
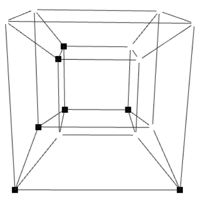
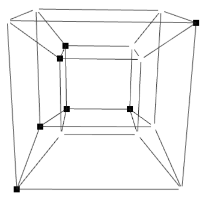
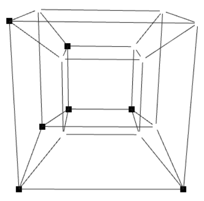
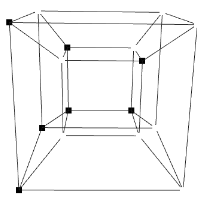
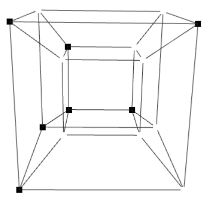
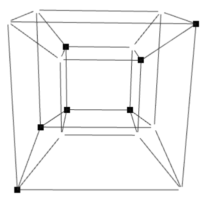
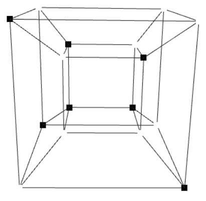
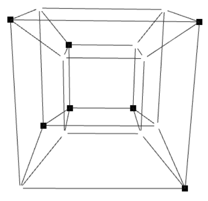
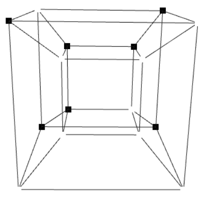
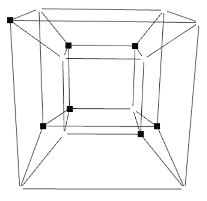
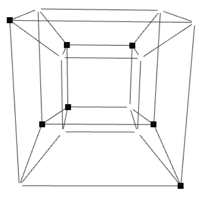
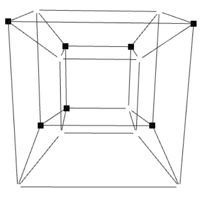
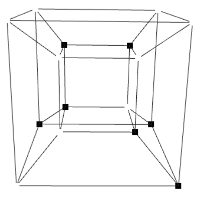
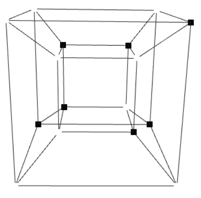


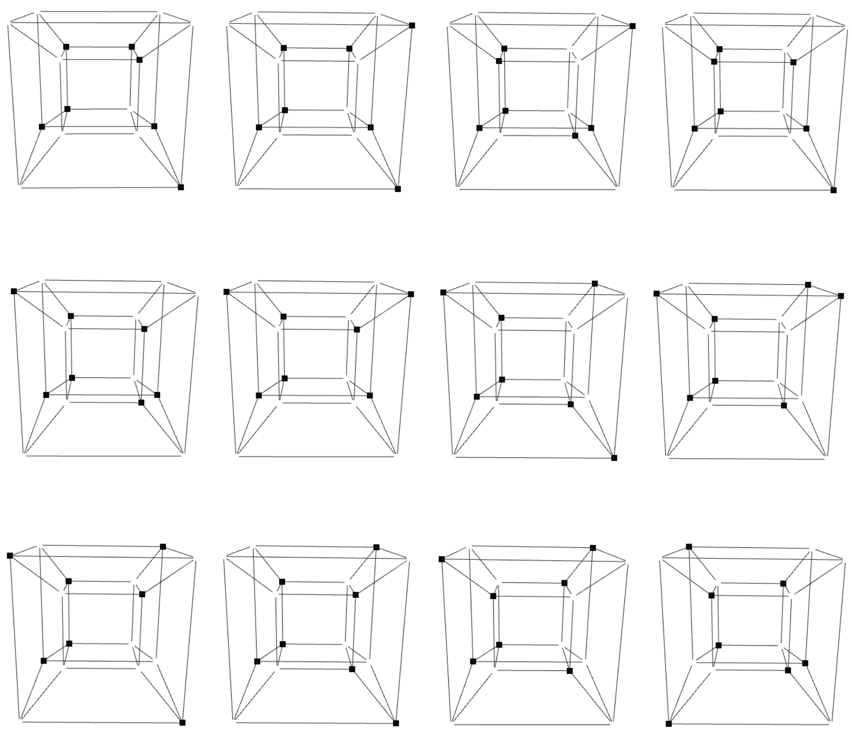




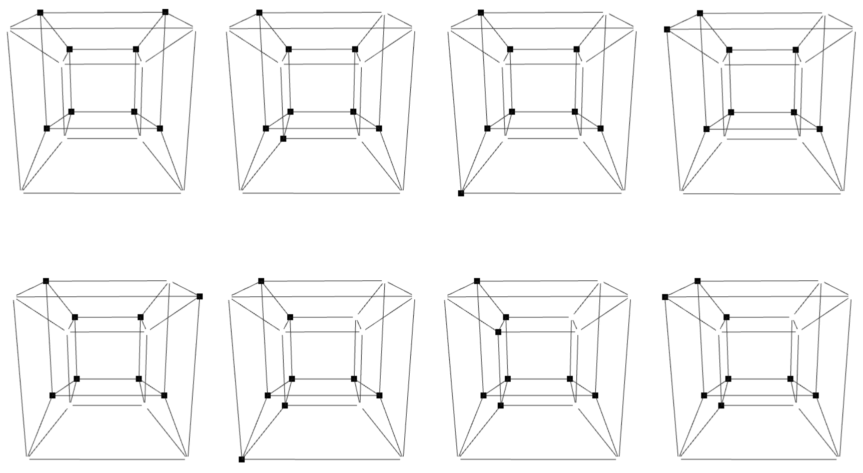
with 7 vertices

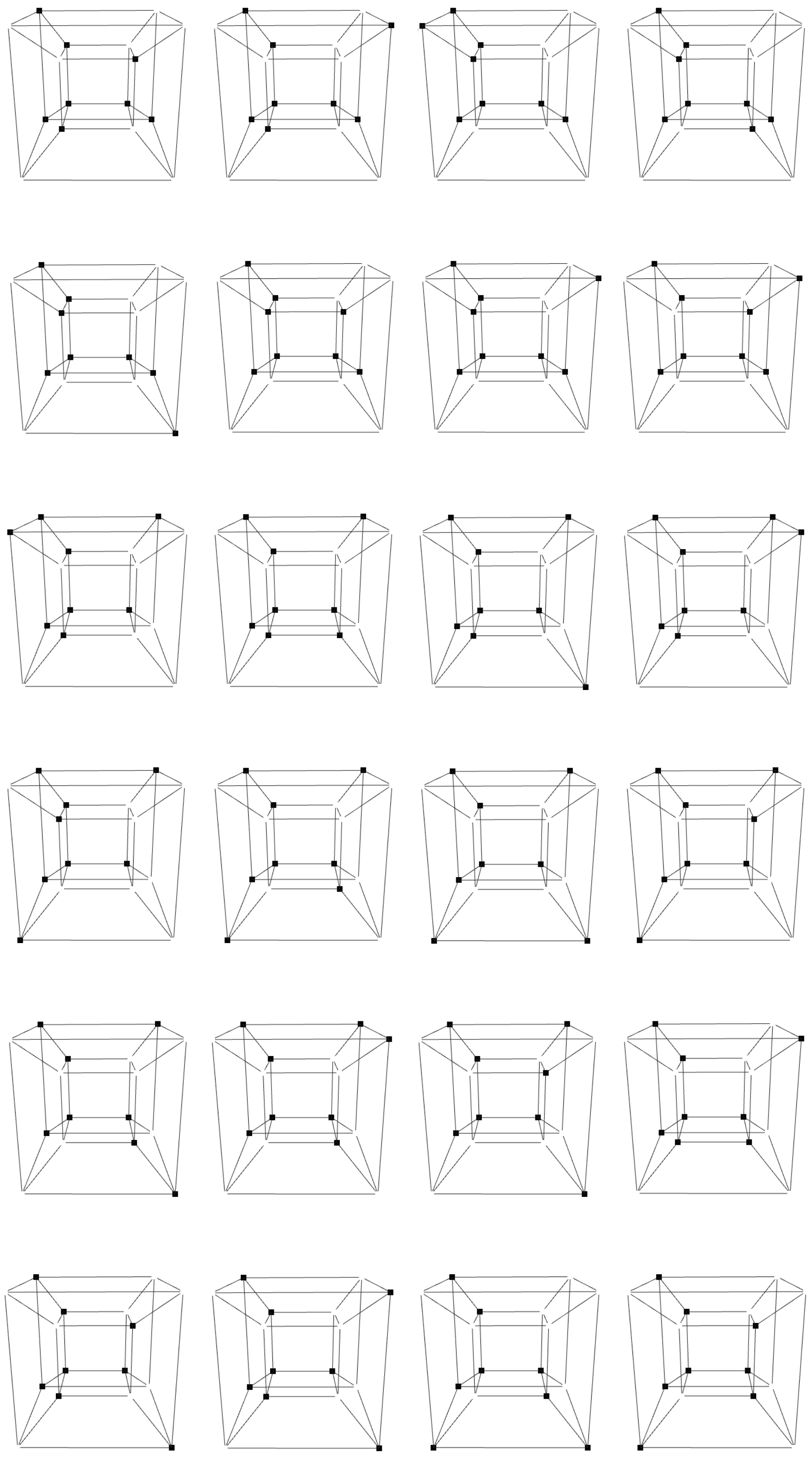


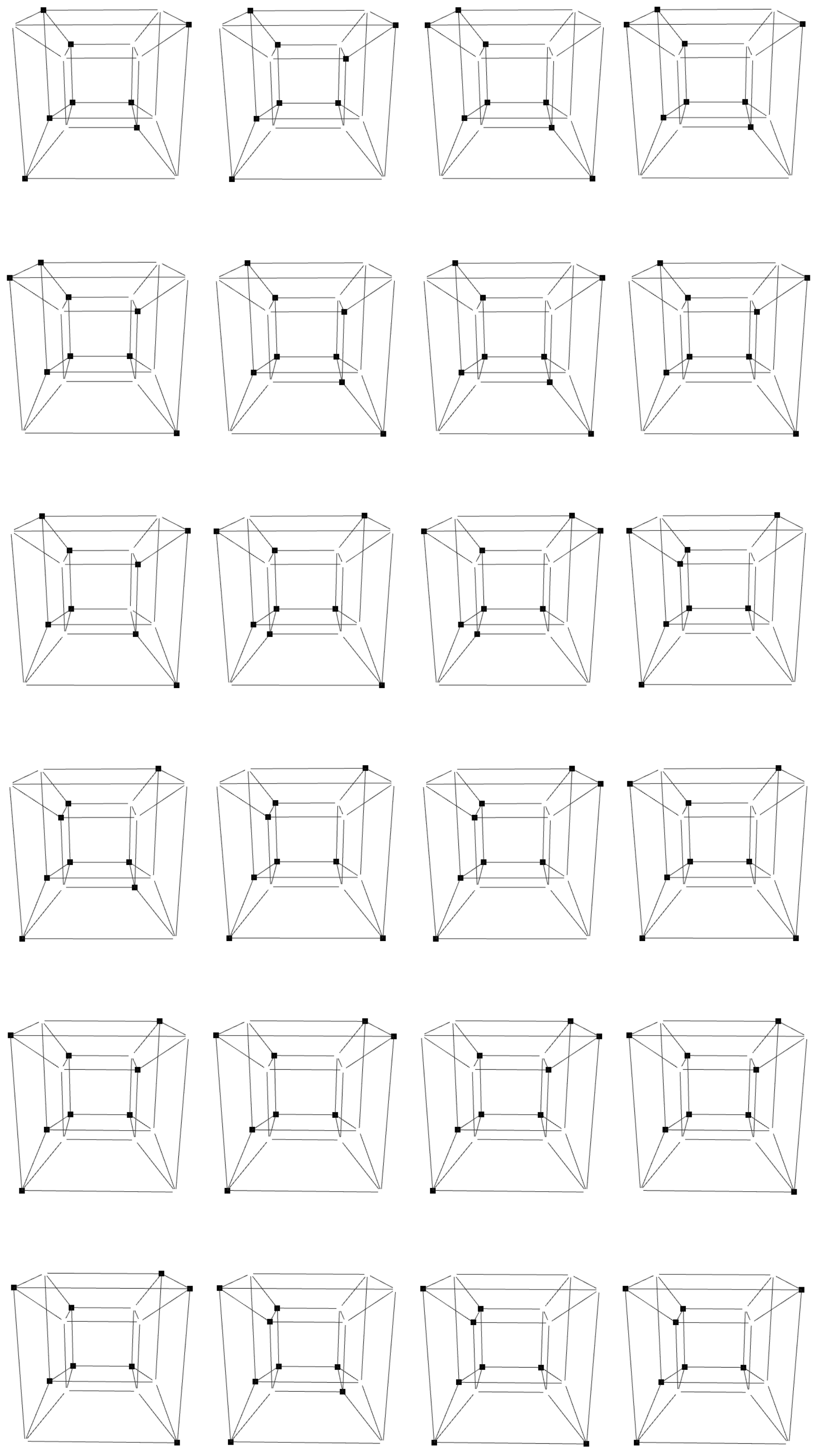

























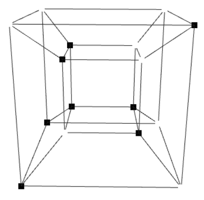
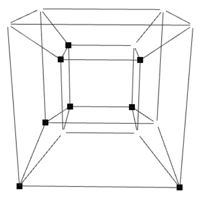
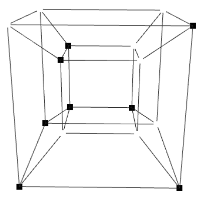
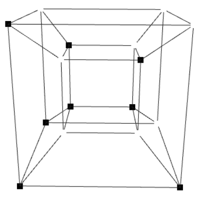
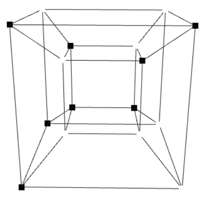
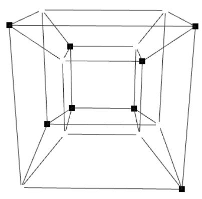
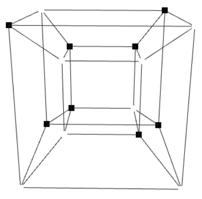
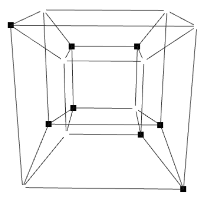
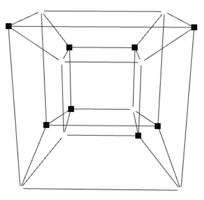
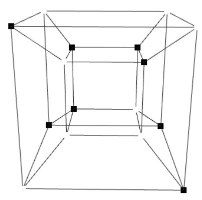
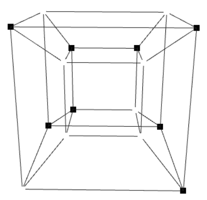
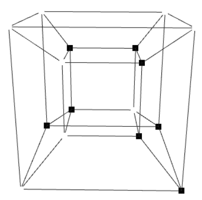
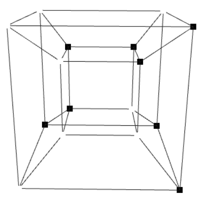
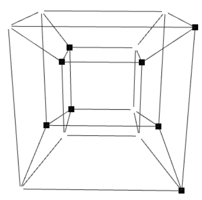
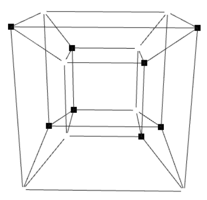
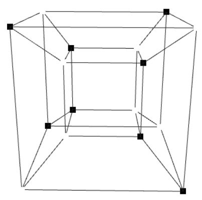
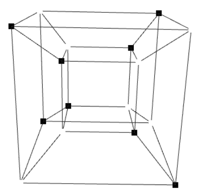
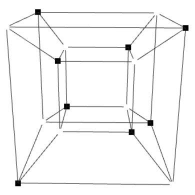


with 8 vertices

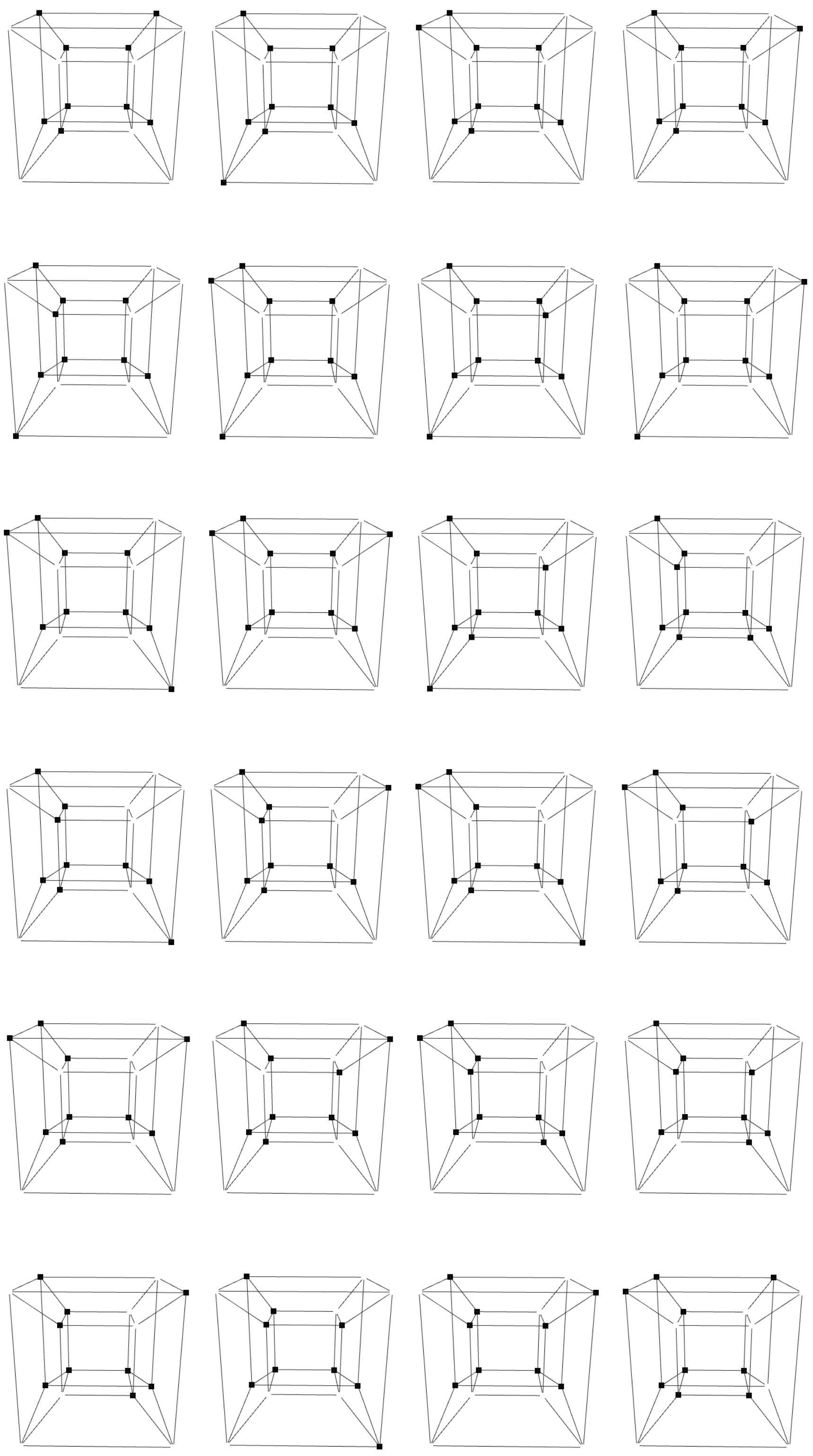


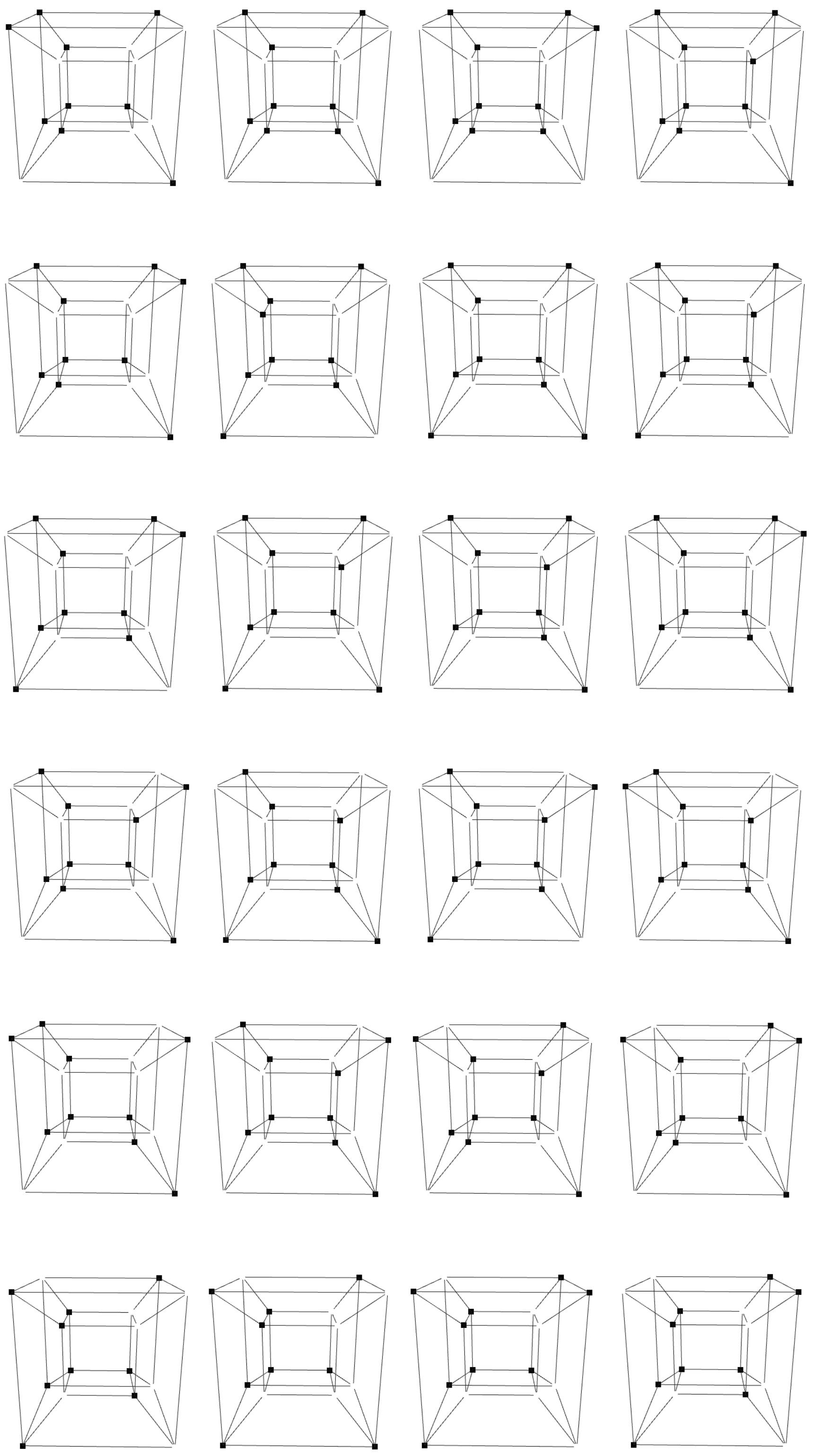


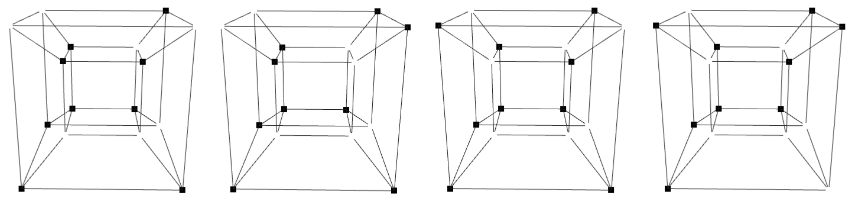


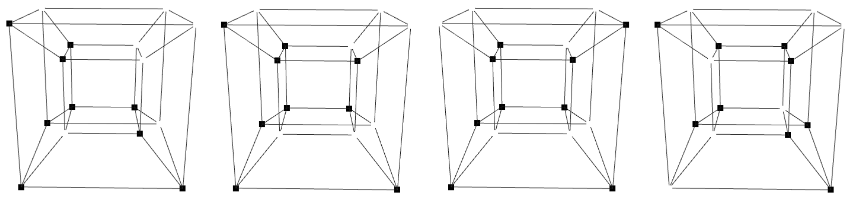



















with 9 vertices

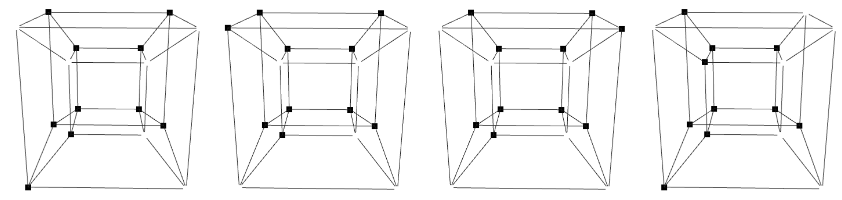


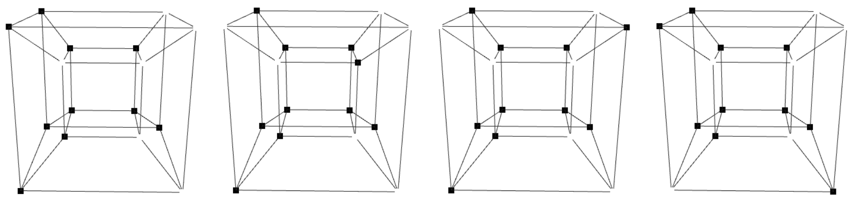


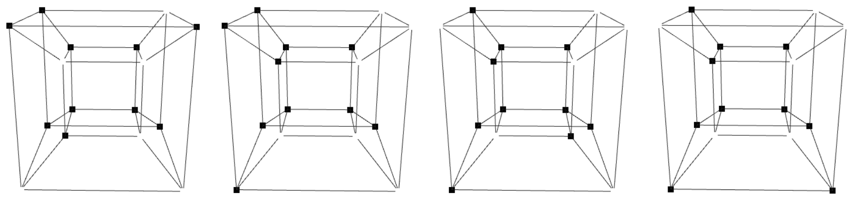


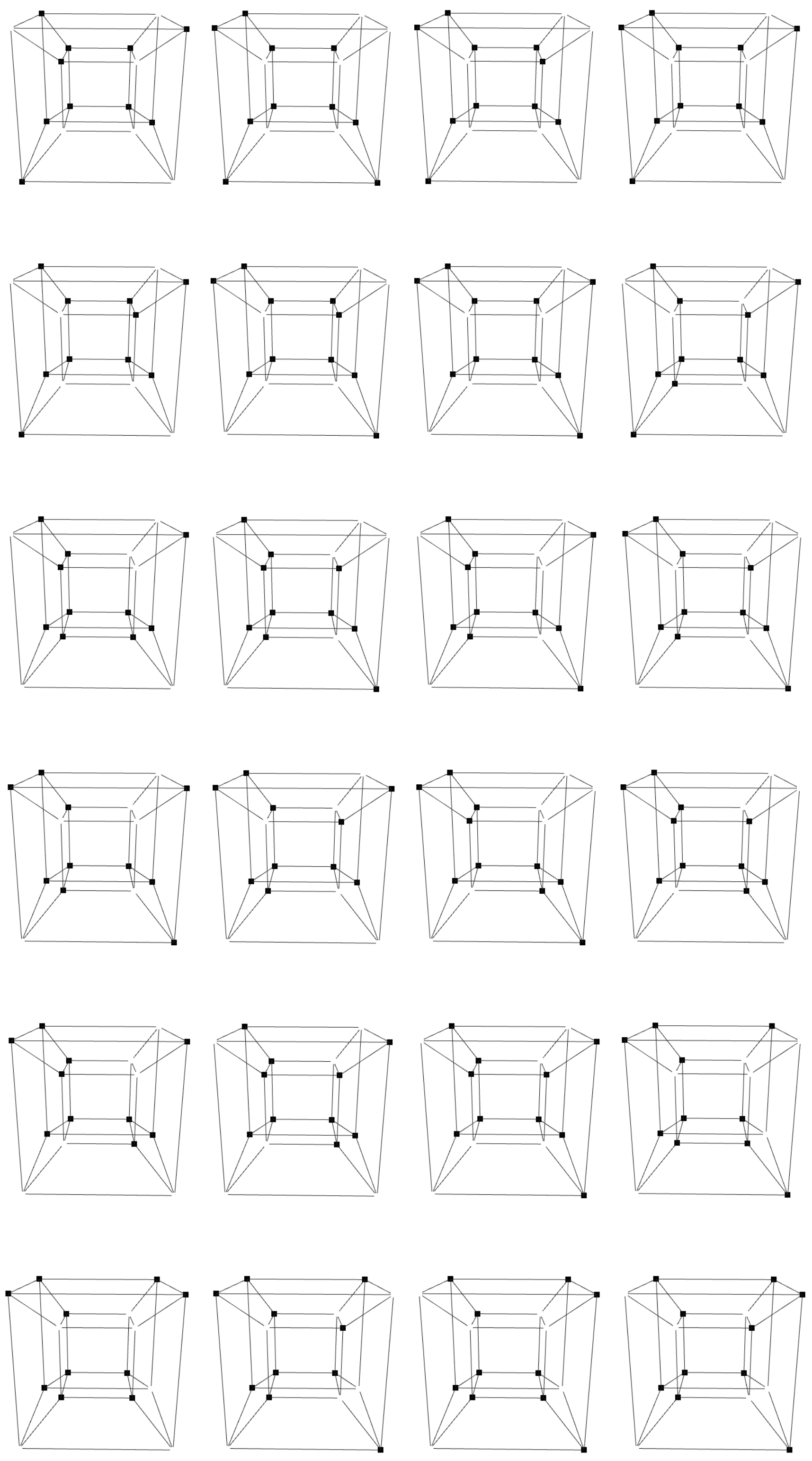


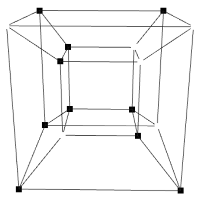
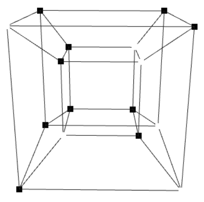
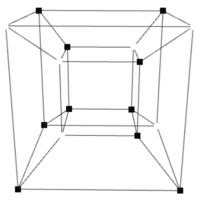
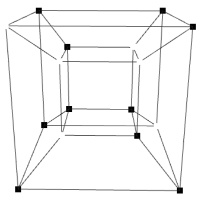
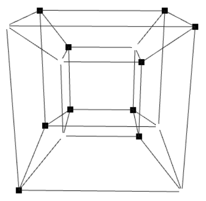
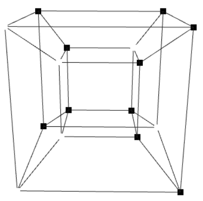
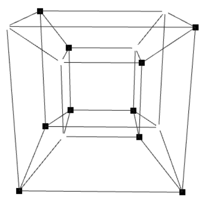
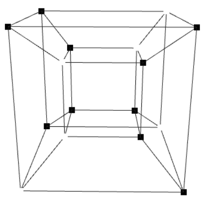
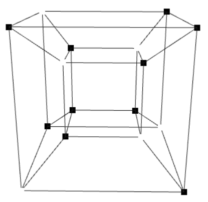
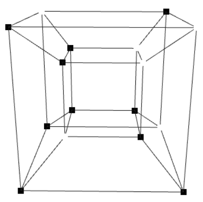
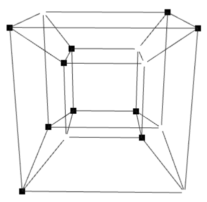
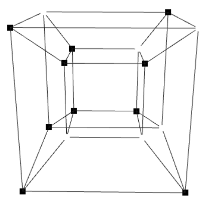
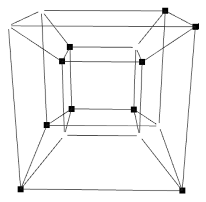
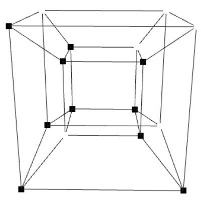
with 10 vertices



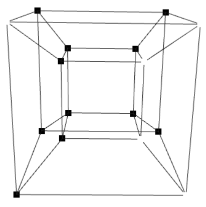
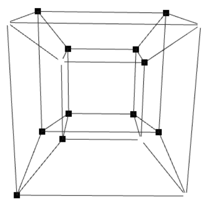
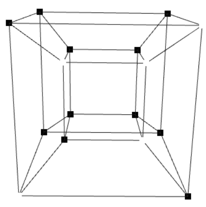
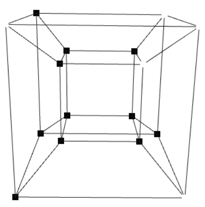


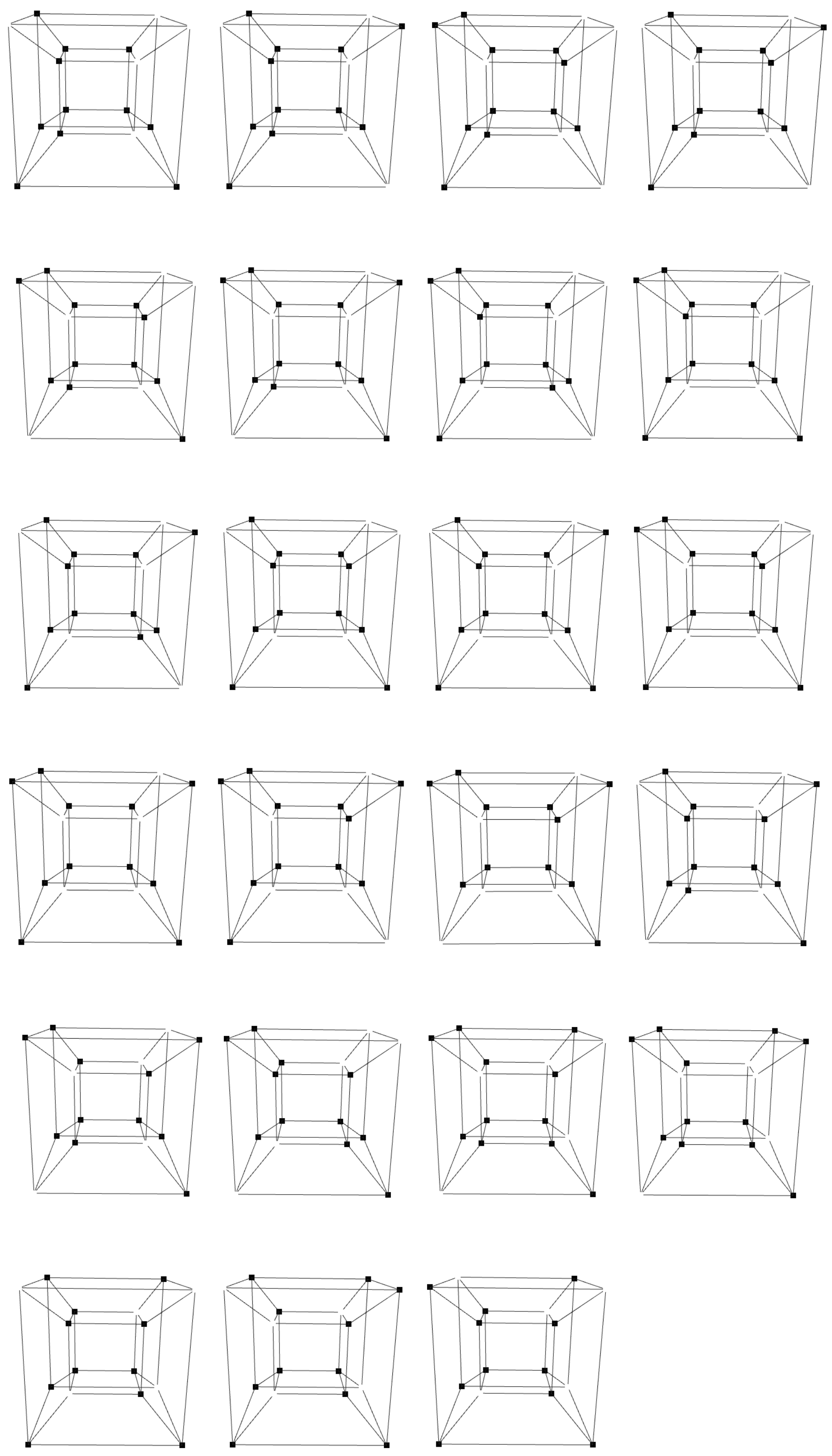




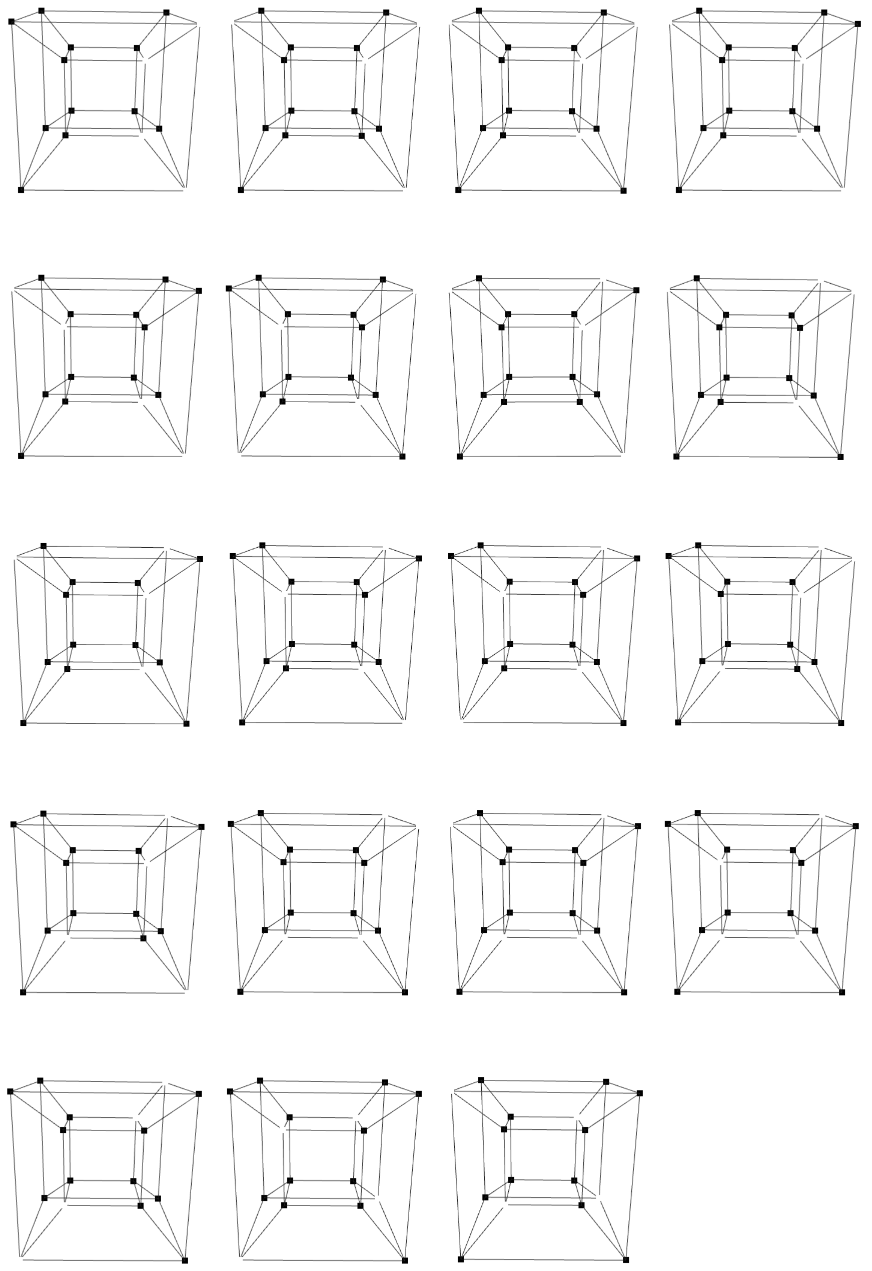















with 11 vertices

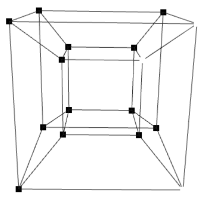
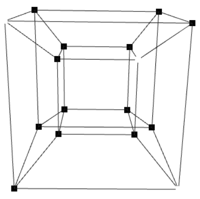
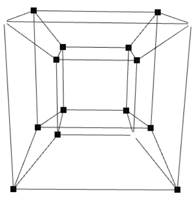
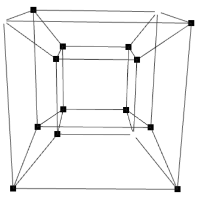
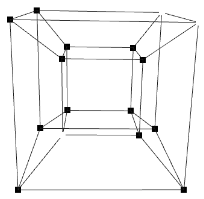
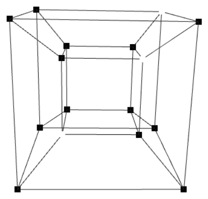







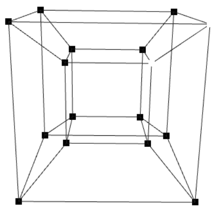
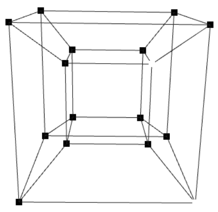
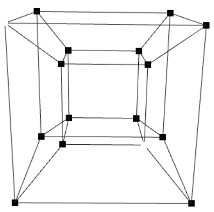
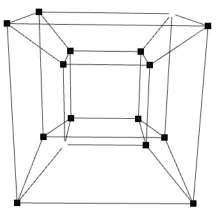
with 12 vertices



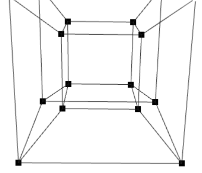
with 13 vertices

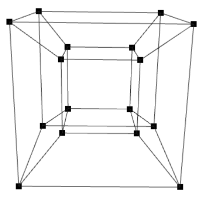
with 14 vertices

with 15 vertices

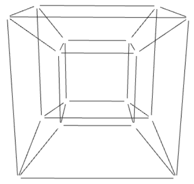


with 16 vertices

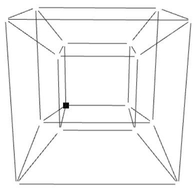


B. PATTERN CELLS IN $\mathbb{R}^4$

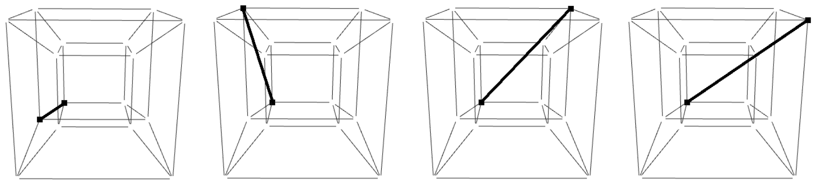
with 0 vertices



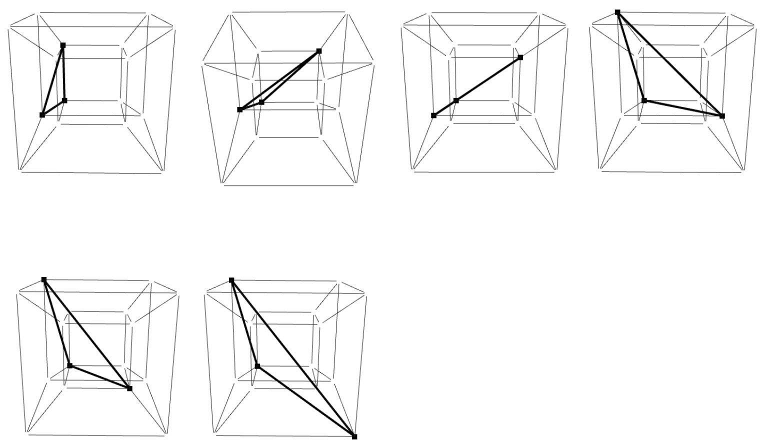
with 1 vertex



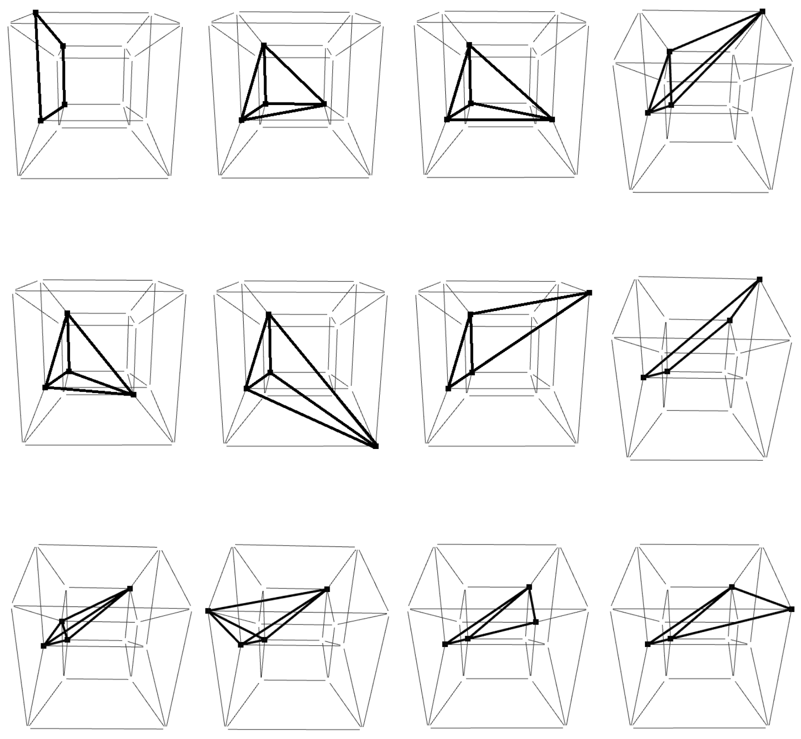
with 2 vertices

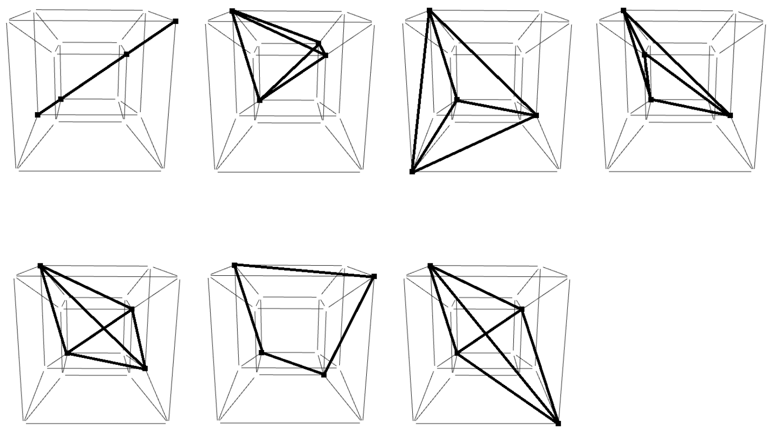


with 3 vertices

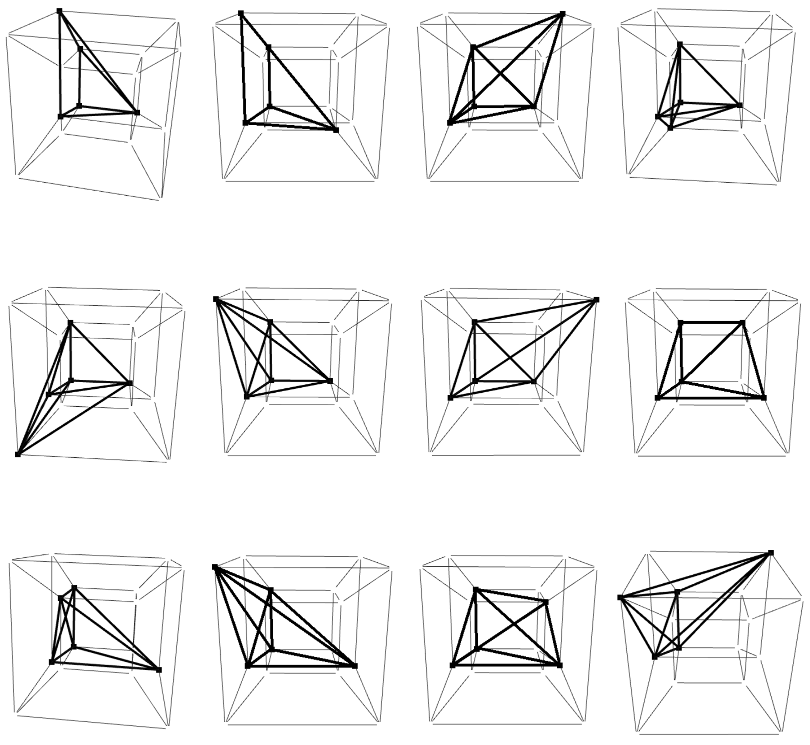


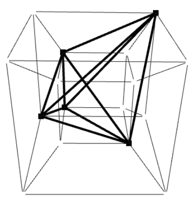
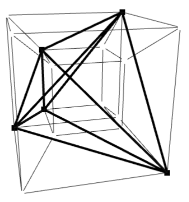
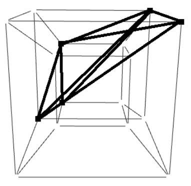
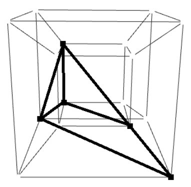
with 4 vertices

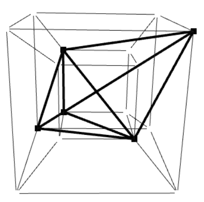
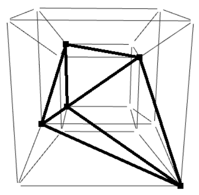
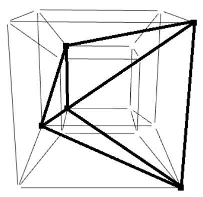
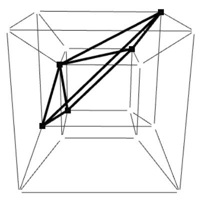


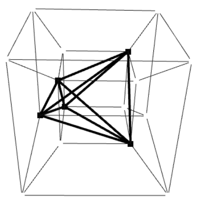
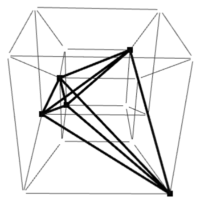
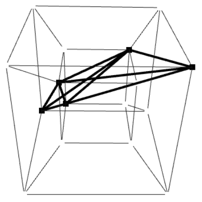
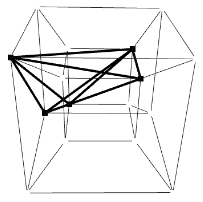


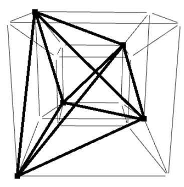
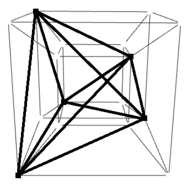
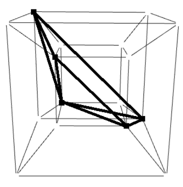
with 5 vertices



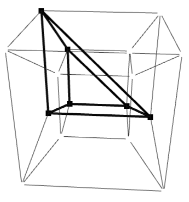
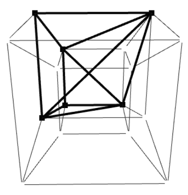
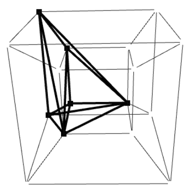
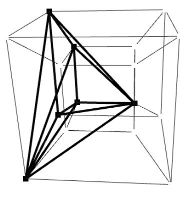





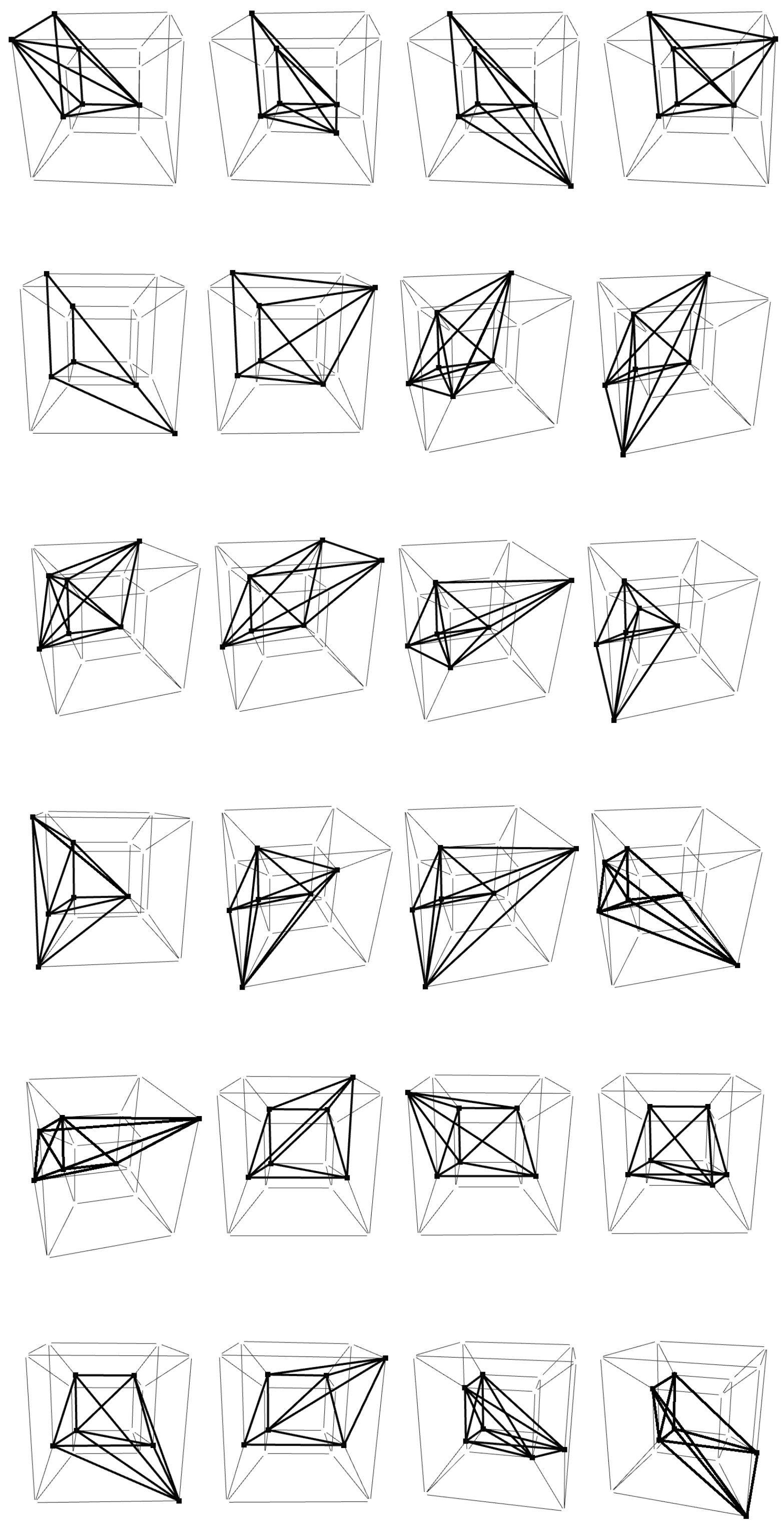





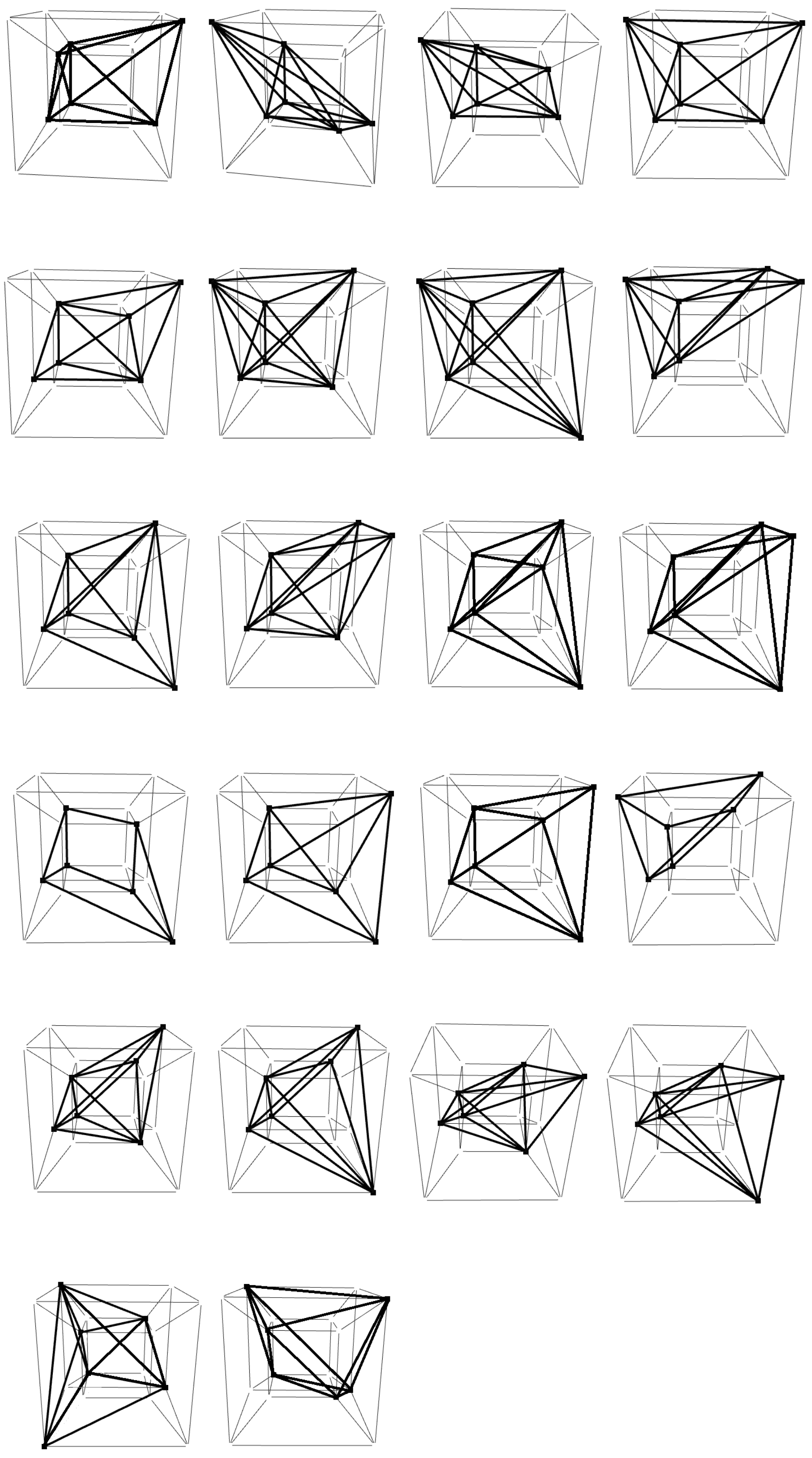





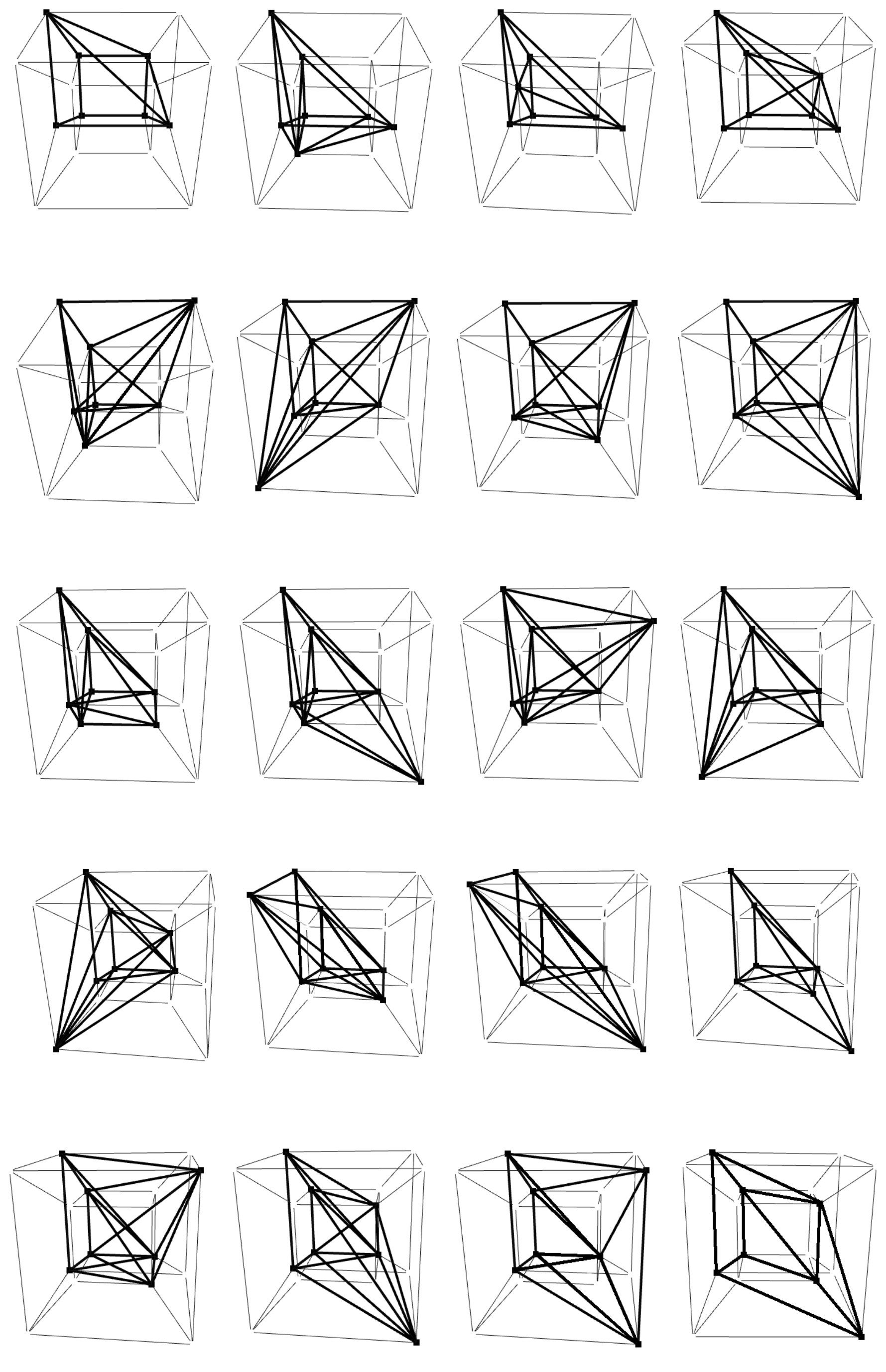
with 6 vertices

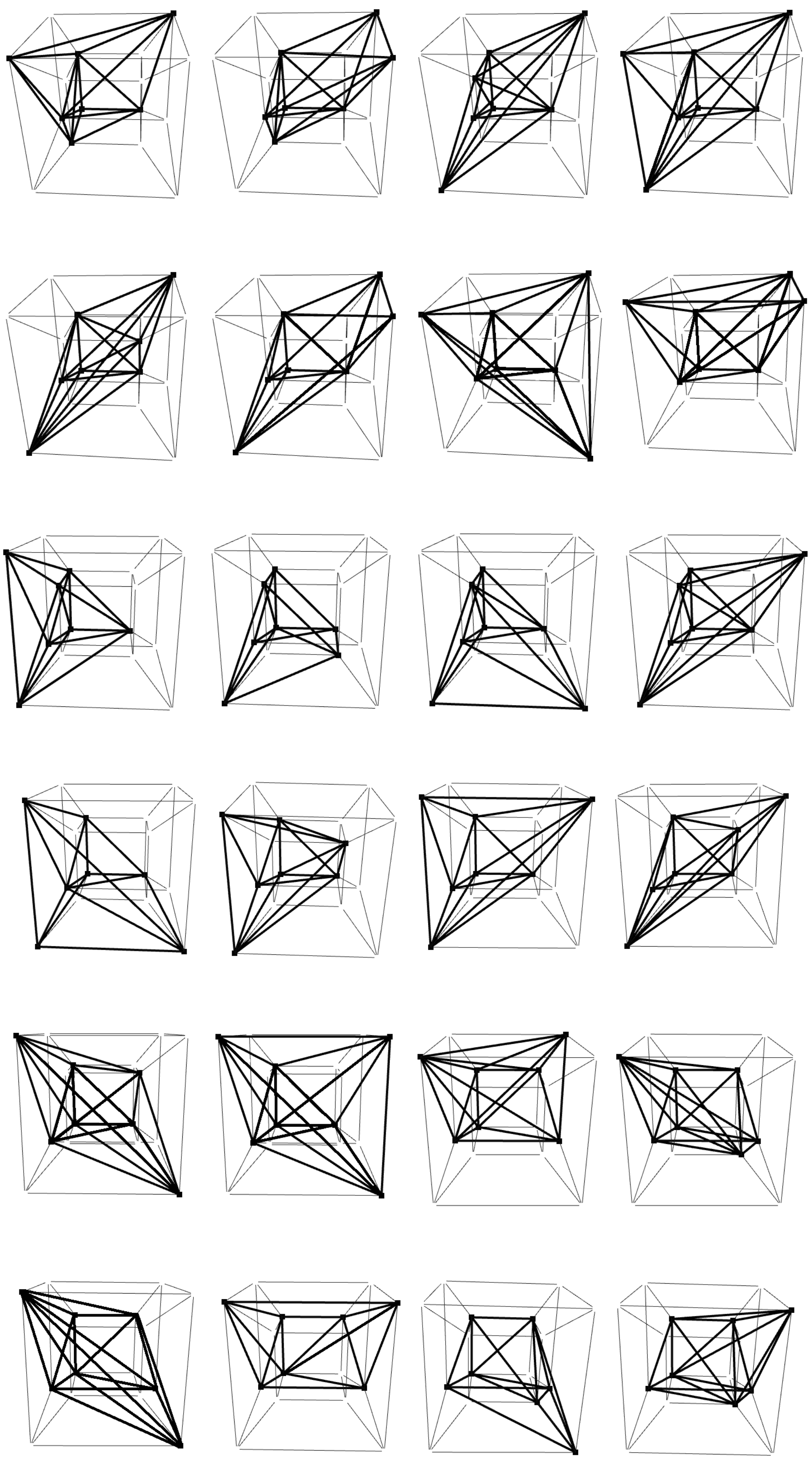





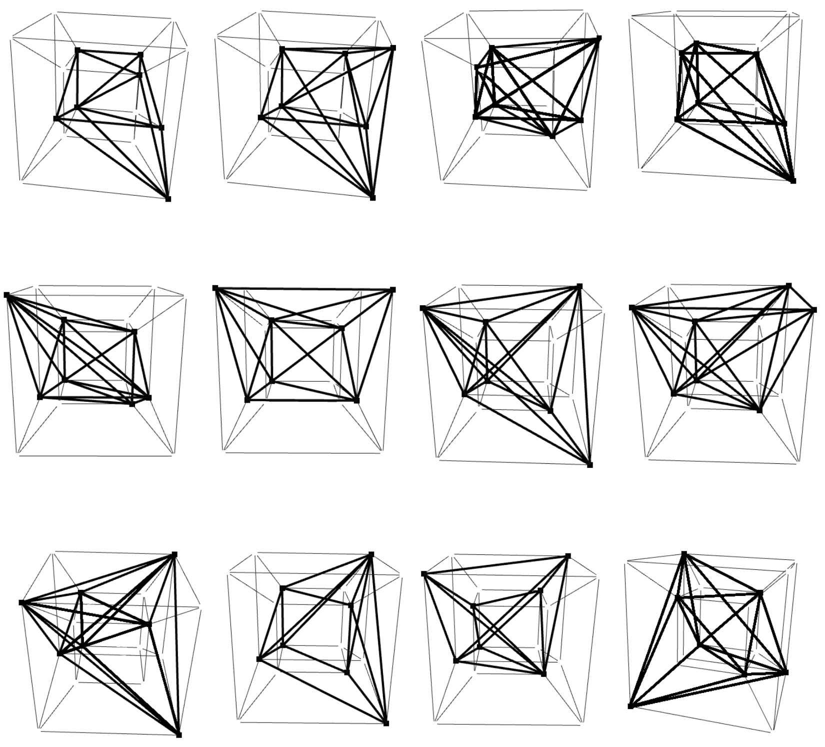




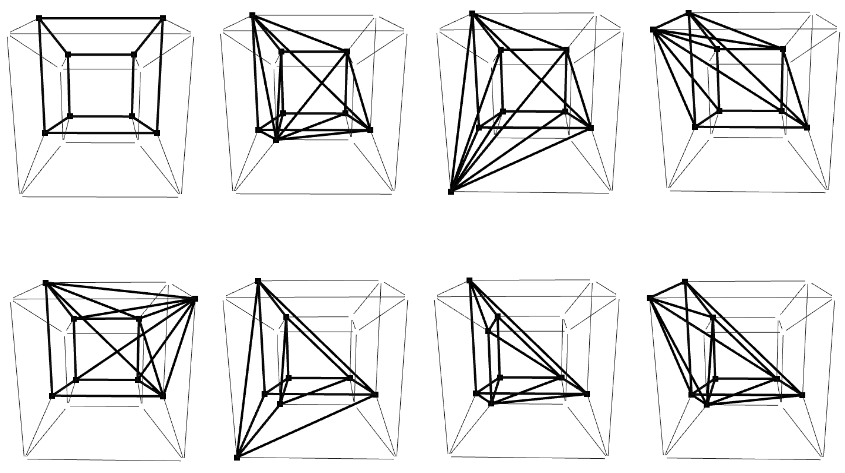
with 7 vertices

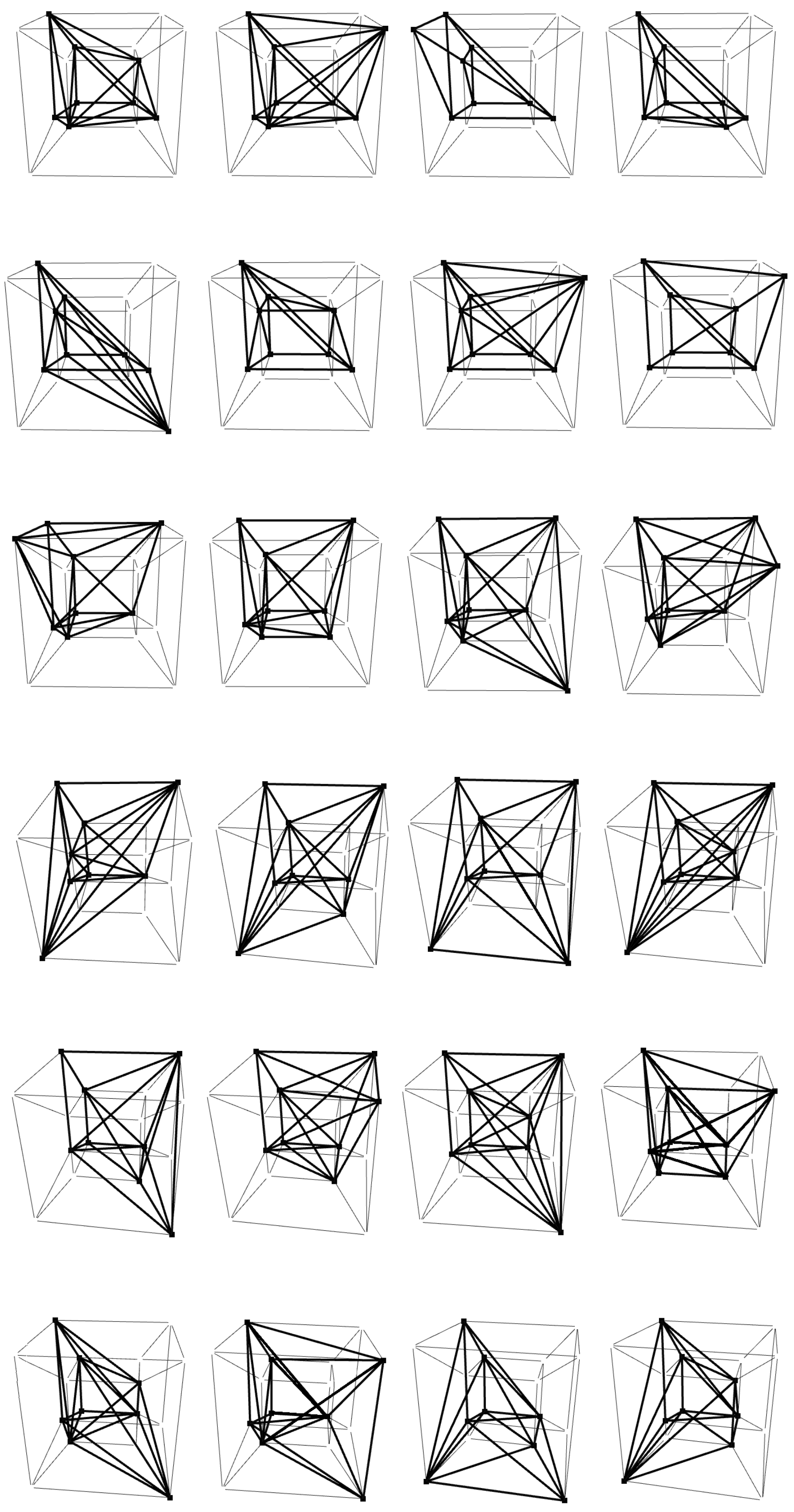


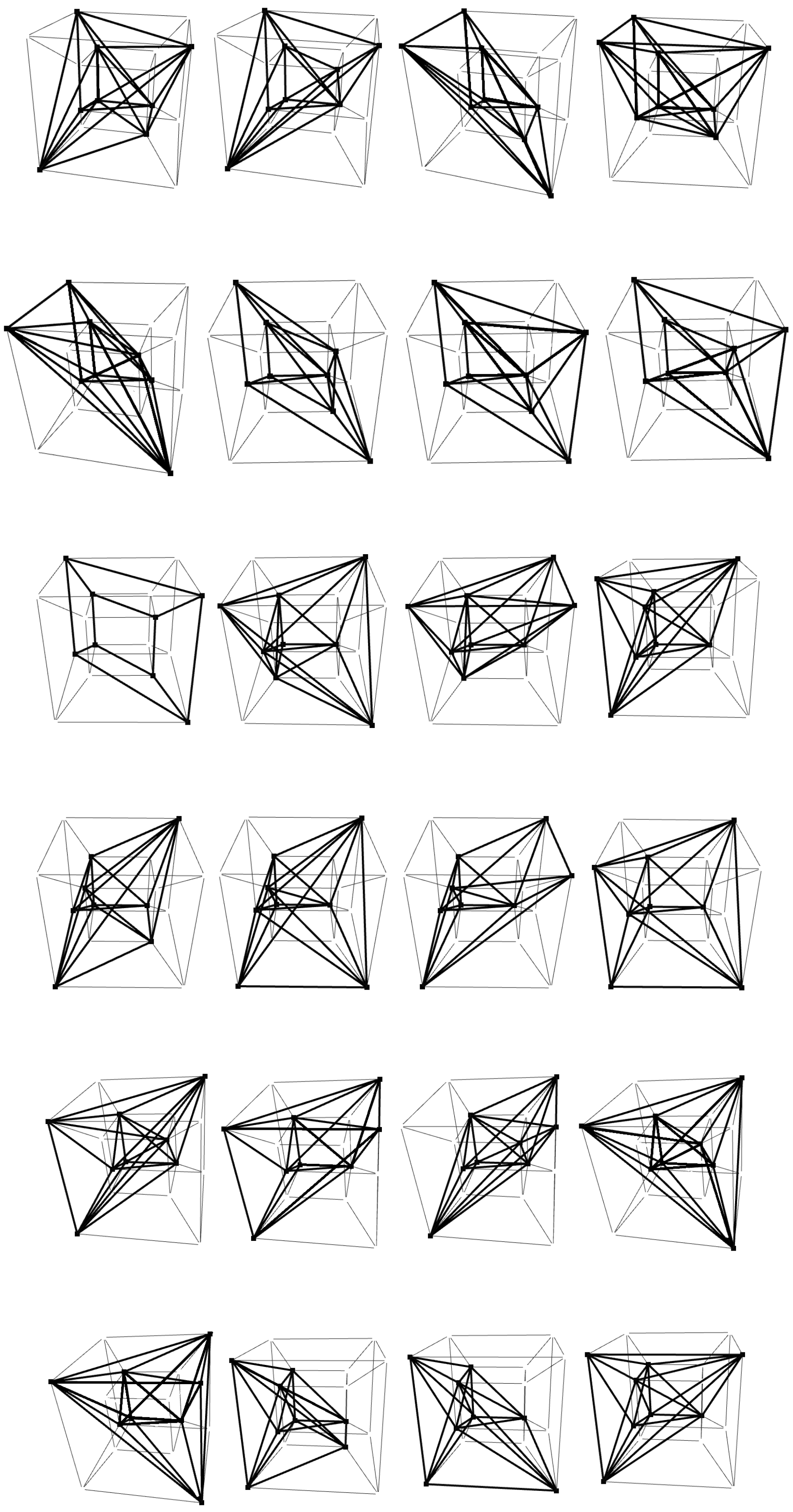


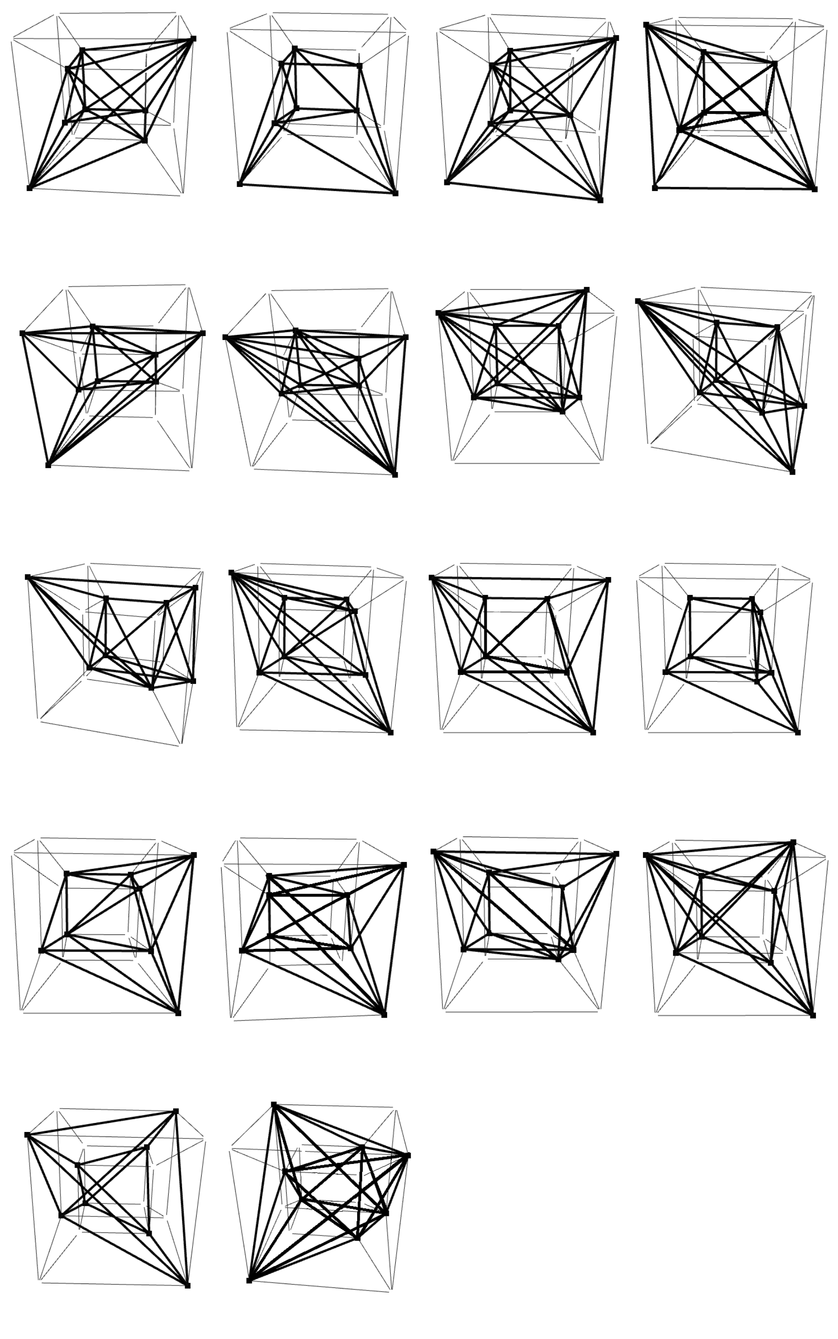


with 8 vertices

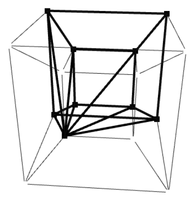
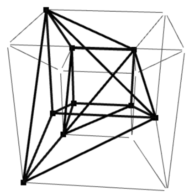
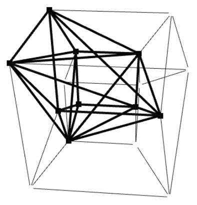
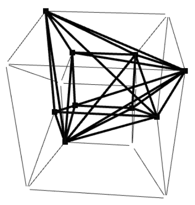
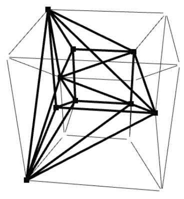
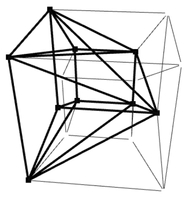
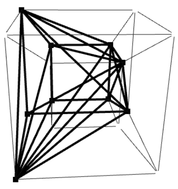
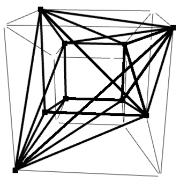
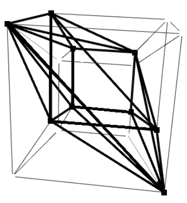
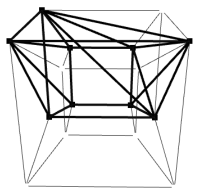
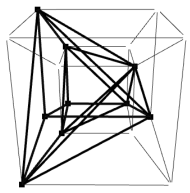
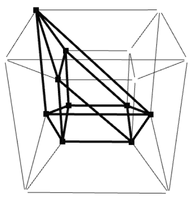
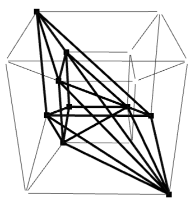
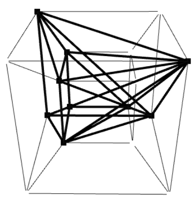
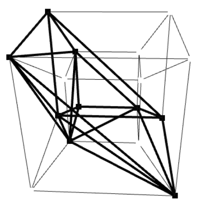
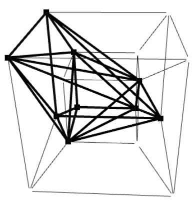
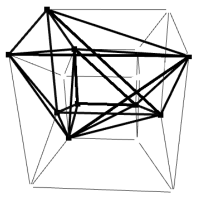
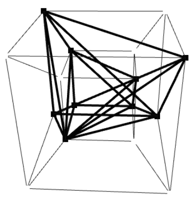
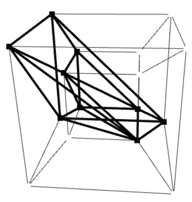
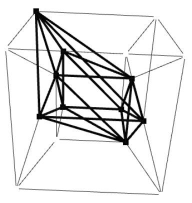
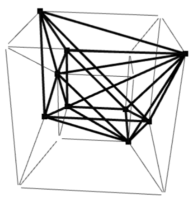
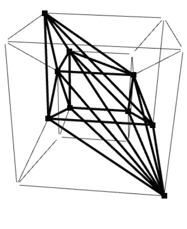
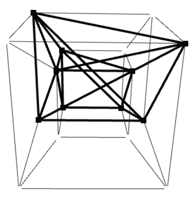
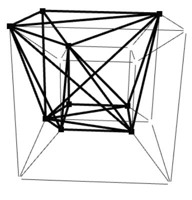


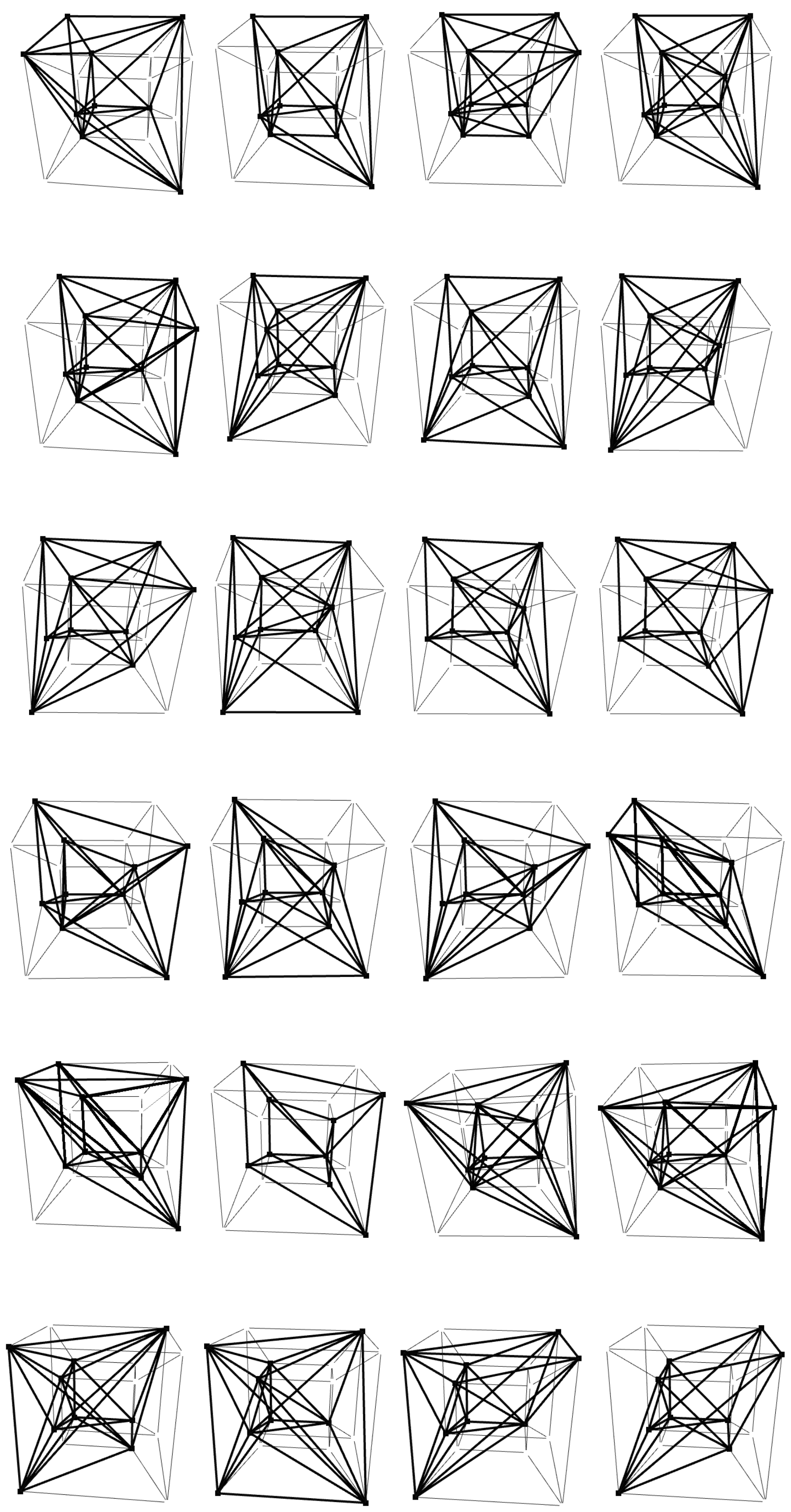


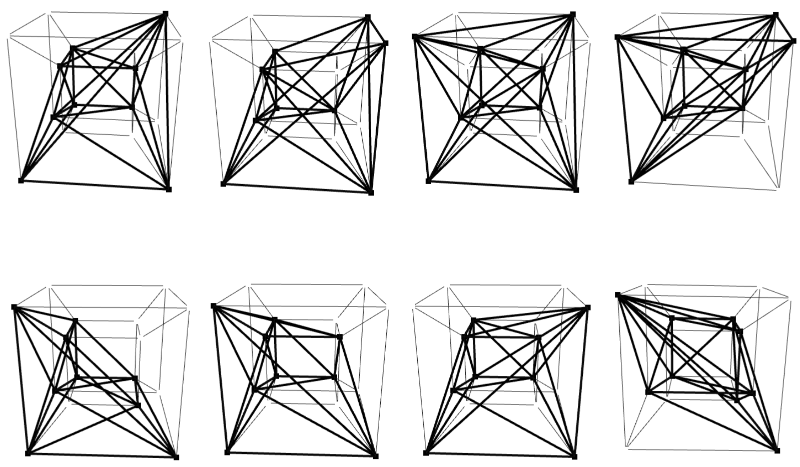




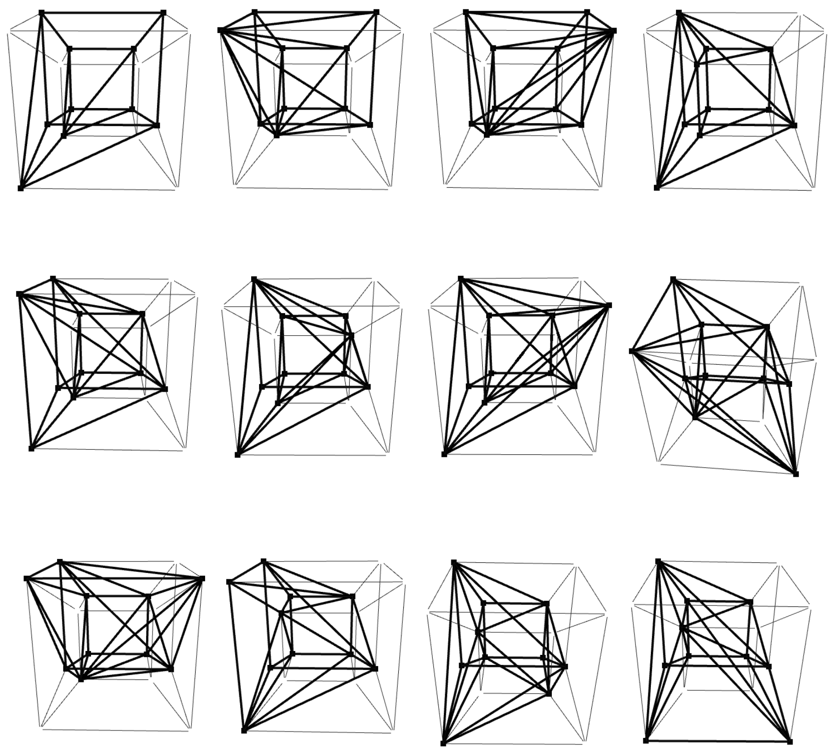
with 9 vertices

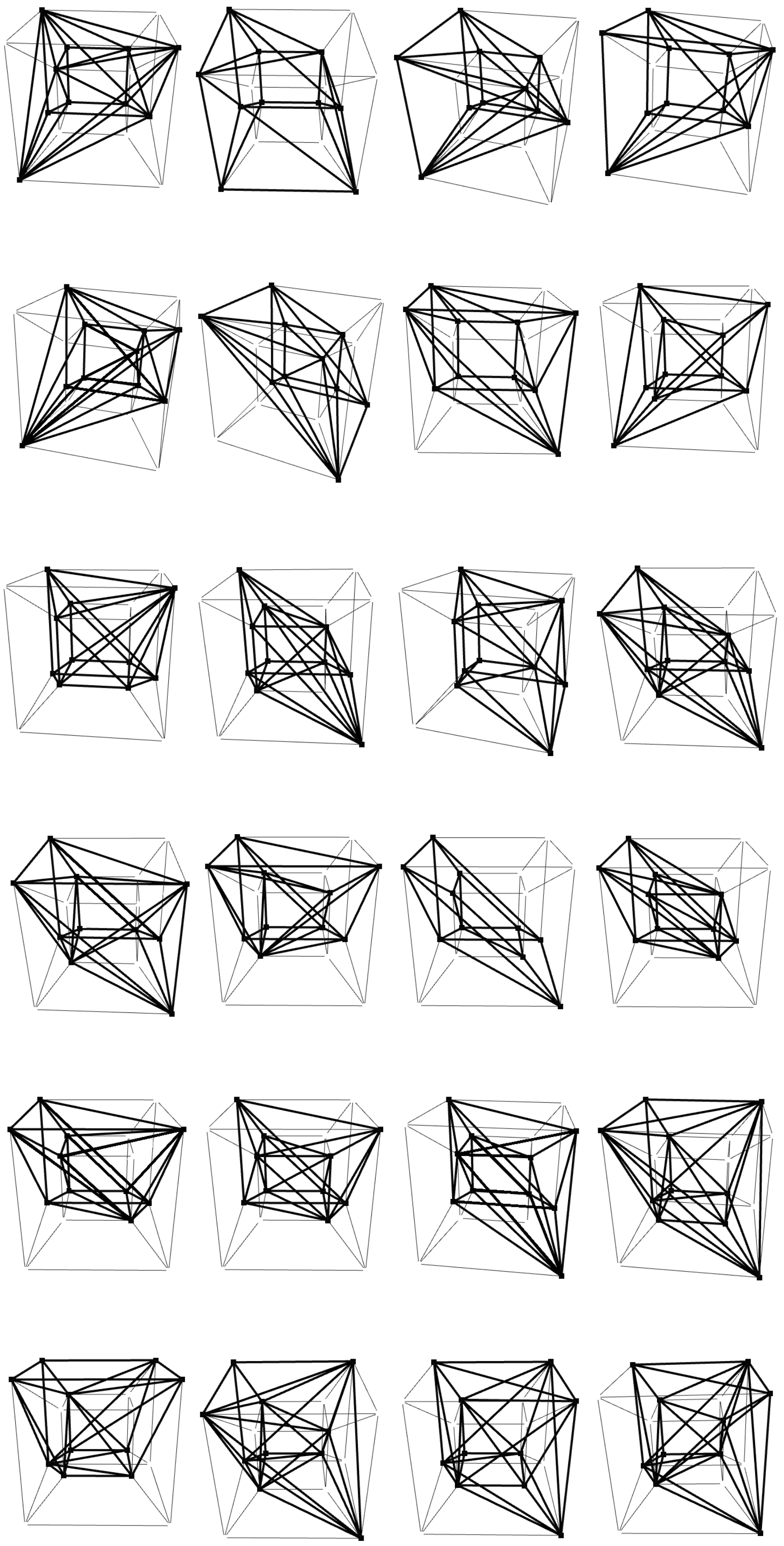

























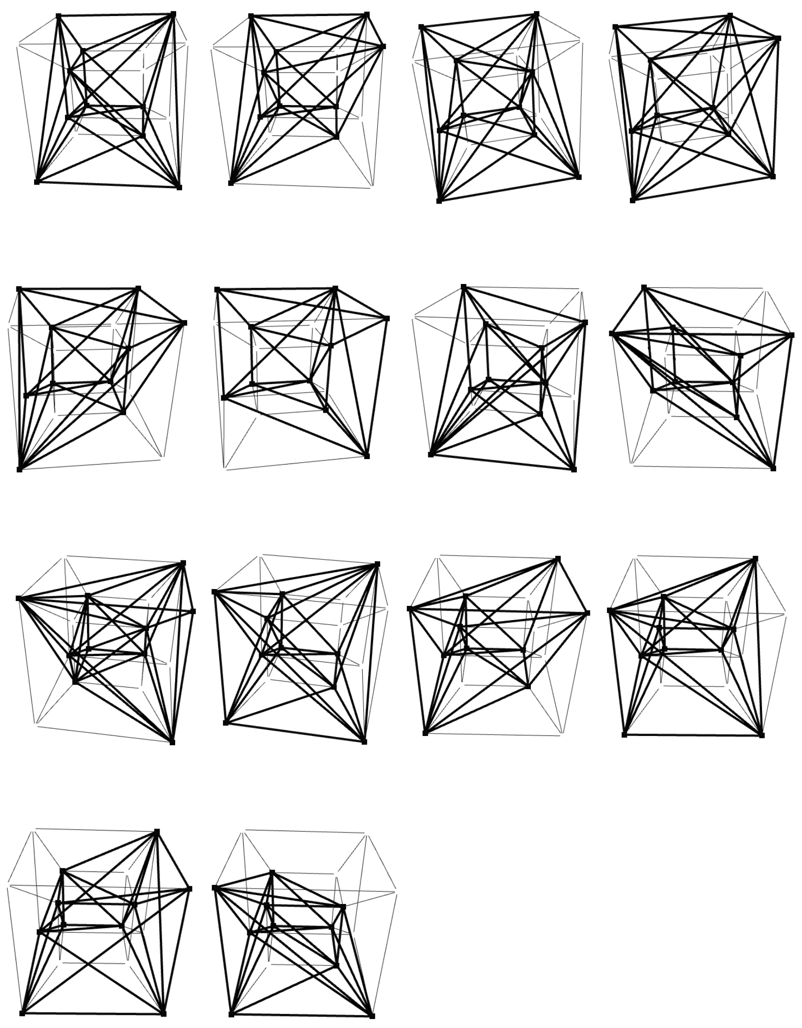




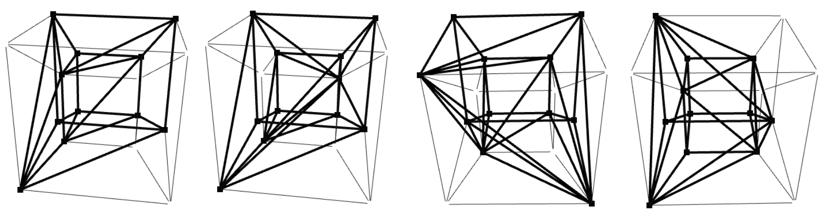
with 10 vertices

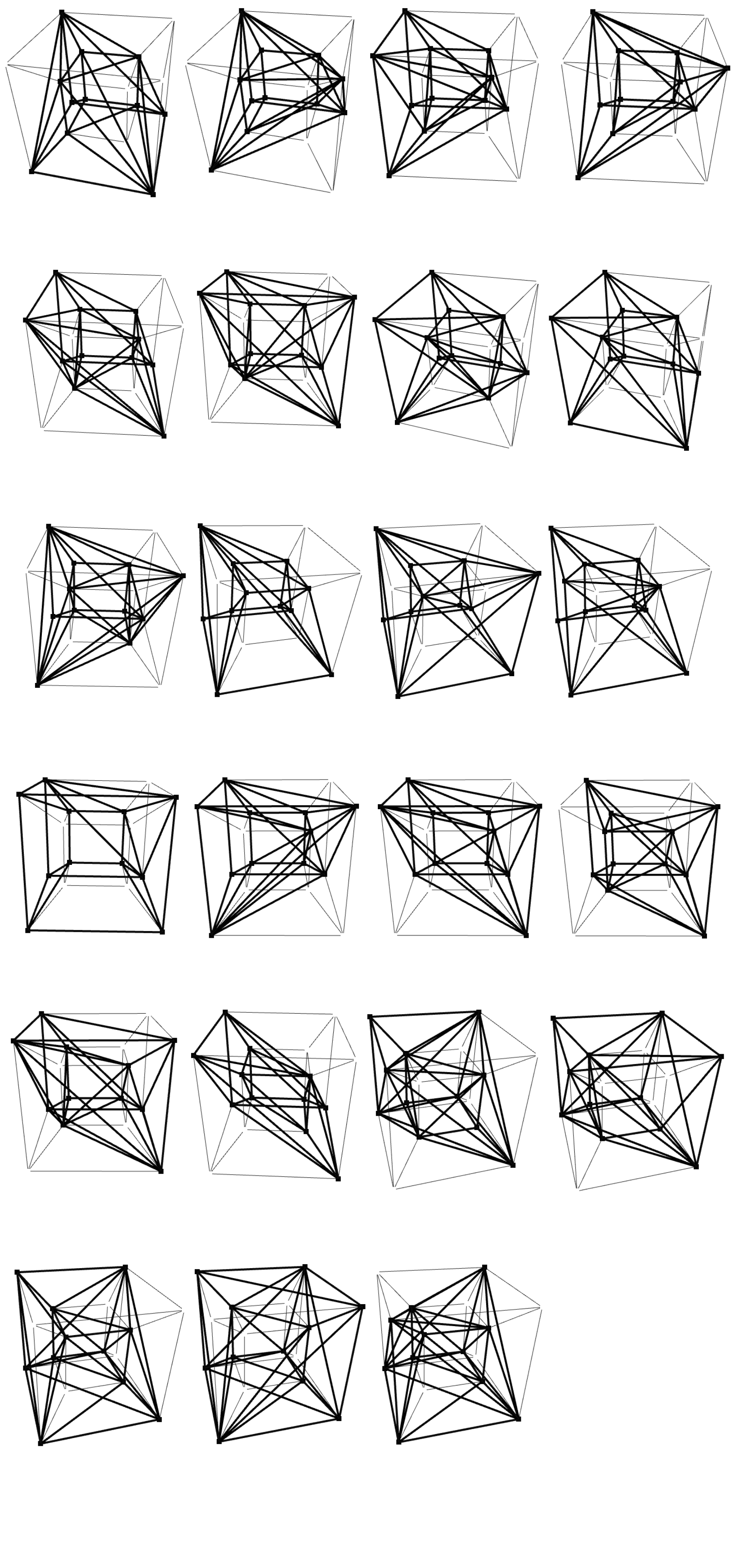




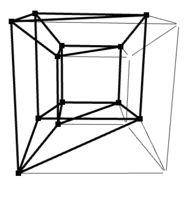
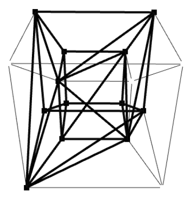
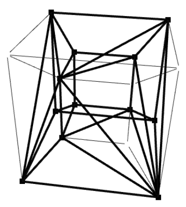
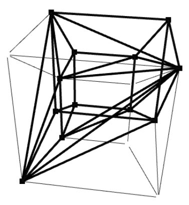
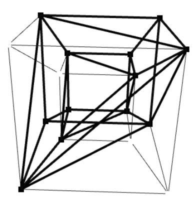
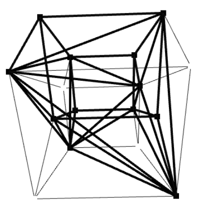
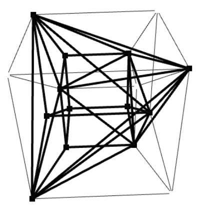
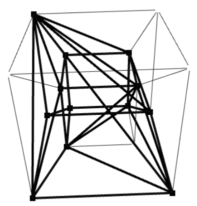
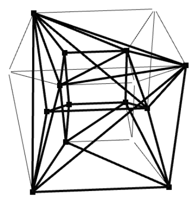
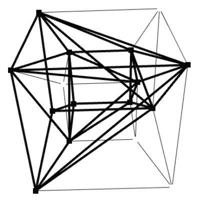
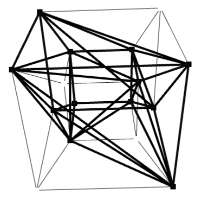
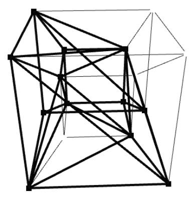
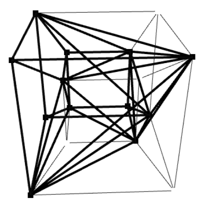
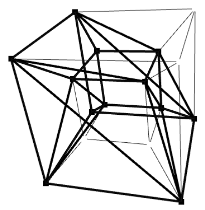
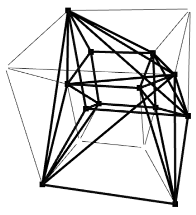
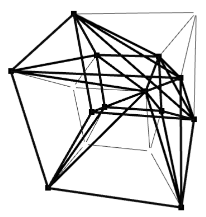
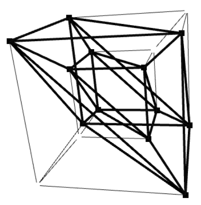
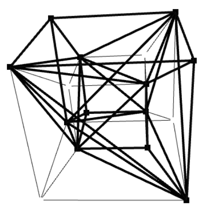
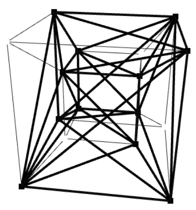


with 11 vertices

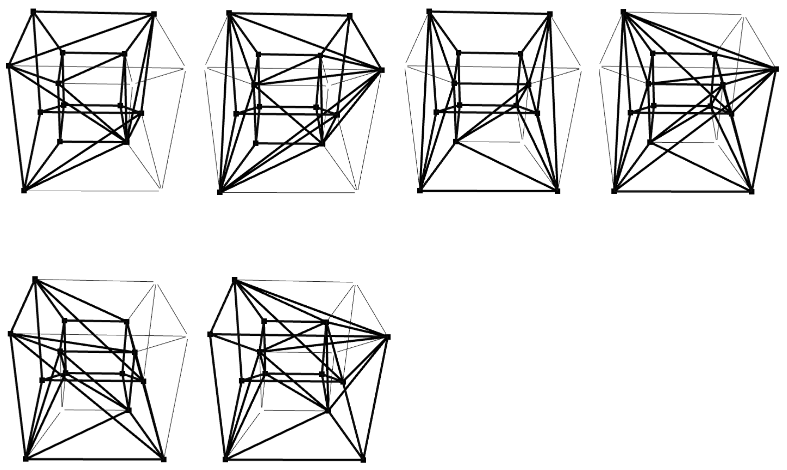




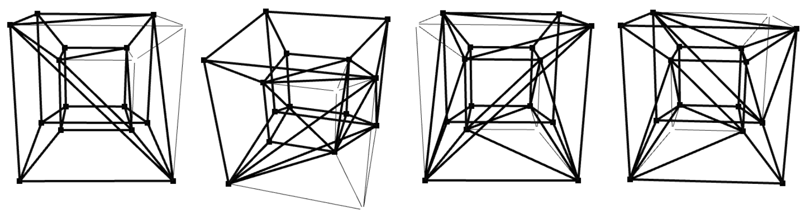
with 12 vertices

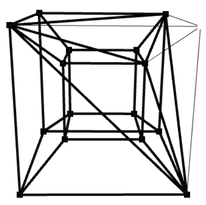
with 13 vertices



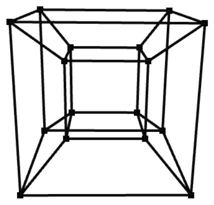
with 14 vertices



with 15 vertices



with 16 vertices



Figures in Appendices A and B are screenshots of “**Hypercube**” software developed by Régis Meysonnier in his Final Master Project supervised by Prof. Jean-Luc Mari from Marseille University (see [29])